Optimizing the Path Towards Plastic-Free Oceans

Dick den Hertog
Amsterdam Business School, University of Amsterdam, Amsterdam, The Netherlands
d.denhertog@uva.nl

Jean Pauphilet
Management Science and Operations, London Business School, London, United Kingdom
jpauphilet@london.edu

Yannick Pham, Bruno Sainte-Rose
The Ocean Cleanup, Rotterdam, The Netherlands
{y.pham,bruno.sainte-rose}@theoceancleanup.com

Baizhi Song
Management Science and Operations, London Business School, London, United Kingdom
bsong@london.edu

Increasing ocean plastic pollution is irreversibly harming ecosystems and human economic activities. We partner with a non-profit organization and use optimization to help clean up oceans from plastic faster. Specifically, we optimize the route of their plastic collection system in the ocean to maximize the quantity of plastic collected over time. We formulate the problem as a longest path problem in a well-structured graph. However, since collection directly impacts future plastic density, the corresponding edge lengths are non-linear polynomials. After analyzing the structural properties of the edge lengths, we propose a search-and-bound method, which leverages a relaxation of the problem solvable via dynamic programming and clustering, to efficiently find high-quality solutions (within 6%-optimal in practice), and develop a tailored branch-and-bound strategy to solve it to provable optimality. On one-year of ocean data, our optimization-based routing approach increases the quantity of plastic collected by over 60% compared with their current routing strategy, hence speeding up the progress towards plastic-free oceans.

Key words: Ocean cleaning; Sustainable operations; Longest path problem; Dynamic programming; Branch-and-bound; Polynomial optimization

1. Introduction

Oceans are vital to life on earth: they home a vast array of plant and animal species and play a critical role in regulating the climate. In addition, they provide important economic benefits, for example by supporting industries like fishing, aquaculture, tourism, and the extraction of minerals. However, oceans are being threatened by growing and severe plastic pollution. As of 2015, 80% of the 6.3 billion tonnes of plastic waste ever generated ended up in landfills or the natural environment (Geyer et al. 2017). As of 2020, there were around 3 million tonnes of plastic waste floating in the oceans (Kaandorp et al. 2023). Furthermore, the amount of plastic emissions in the ocean increases by 4% every year (Kaandorp et al. 2023), with 0.5–2.7 million tonnes emitted via rivers every year (Meijer et al. 2021, Lebreton et al. 2017, Schmidt et al. 2017). Plastic pollution is posing a threat to the marine ecosystem and the species that rely on it (Gall and Thompson 2015, Wilcox et al. 2015). It also has a detrimental impact on human activities. We refer to Li et al. (2016) for a comprehensive review of marine plastic pollution, its sources, and effects. Because of its environmental and economic relevance, the reduction of oceans pollution has been listed as an explicit target in the United Nations' Sustainable Development Goal #14 'Life Below Water'.

The reduction of marine plastic pollution requires two concurrent actions: reducing yearly emissions and removing persistent legacy plastic pollution. Regarding the first effort, many legislative and non-legislative actions have been taken to ban (or discourage) the use of single-use plastic (e.g., plastic bags or straws), with varying degrees of efficiency (see Schnurr et al. 2018, for a review). Given the importance of land-based pollution and the role of rivers in transporting land-based pollution into the oceans, solutions also include improved in-land plastic waste management, recycling, and plastic interception in rivers (see, e.g., Dijkstra et al. 2021, Winterstetter et al. 2021). On the other hand, the active removal of plastic already emitted in the oceans has received lower attention and may be regarded as less efficient than preventing emissions due to the low average concentration of floating plastic in the oceans.

Fortunately, floating plastic debris get trapped in large circulating currents, called gyres, and tend to accumulate in specific areas called 'garbage patches'. The largest of these five patches, the "Great Pacific Garbage Patch" (GPGP), is located halfway between California and Hawaii. Nearly 80,000 tonnes of plastic float inside this area of 1.6 million km^2 or three times the size of France (Lebreton et al. 2018). Figure 1 displays yearly average plastic density estimates in the GPGP. In short, plastic density in the GPGP is about 20 times higher than in the rest of the ocean.

The Ocean Cleanup is a Dutch NGO whose mission is to clean up oceans from plastic. In addition to interception activities in rivers, they have developed a technology to collect plastic debris in the oceans. They have been trialling their solutions in the GPGP since 2018, and operating their newest system since 2021. Their system consists of a large (600-meter wide and 4-meter deep, at the beginning of our collaboration) U-shaped screen, slowly dragged by two ships, that can capture floating plastics without capturing any marine animals. In this collaboration, we investigate the potential for improving the efficiency of their plastic collection system by optimizing its route in the GPGP. In particular, we use data and models about weather conditions and plastic density

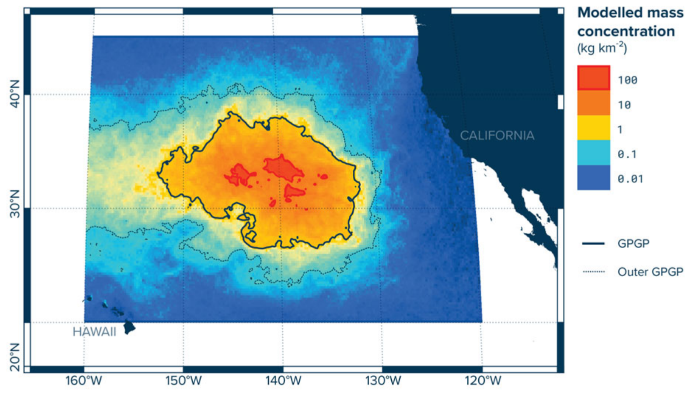


Figure 1 Yearly plastic density map in the GPGP.

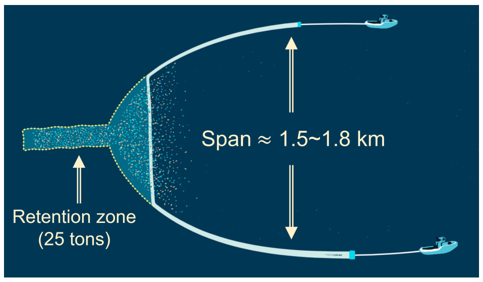


Figure 2 The Ocean Cleanup's system.

in the GPGP to construct an optimization-based routing algorithm that directly maximizes the quantity of plastic collected, hence speeding up the progress towards cleaner and healthier oceans.

1.1. Problem description

The Ocean Cleanup's plastic collection system (which we later refer to as the "system") is composed of two ships and a U-shaped screen (or net), as shown in Figure 2. In its original configuration (system 002/B), the screen had a span of 600 meters, but it has been increased to 1.4 km in the latest version of the system (system 03). It acts like an artificial coastline that intercepts floating debris. These floating plastic particles then gradually accumulate in a partially closed contraption at the apex of the screen, called the retention zone.

Routing the system in the GPGP needs to satisfy some navigation requirements. To preserve the physical integrity of the screen, for example, the system can only slowly change course and cannot make any sharp turns. In addition, in order not to catch any fish or other marine life, the system moves at a fixed and low speed of around 1.5 knots (2.78 km/h). Finally, as any sea vessel, it is sensitive to weather and navigation conditions such as waves and wind. For example, when the wave height exceeds 4.5 meters, the system has to head against the waves to protect the screen. Above 6-meter waves, the screen no longer intercepts any plastic.

The retention zone has a limited capacity of 25 metric tons and needs to be emptied regularly. The process of emptying the retention zone is called an extraction and is a complex operation: the screen is closed, the retention zone is hauled onto the ship deck and lifted by a crane, and the collected plastic is discharged on deck, before being sorted and sent on-shore for recycling. In particular, the crane cannot be operated when the wave height exceeds 2.5 meters. Overall, an extraction takes around 24 hours, during which the collection is stopped. Hence, extractions play an important role in the overall collection efficiency and extraction scheduling should be incorporated in our search for a better routing system.

Our primary objective is to maximize the quantity of plastic collected. The Ocean Cleanup has developed a suite of models to estimate the dispersion and density of plastic in the GPGP (Klink et al. 2022), by using hindcast and forecast models of ocean currents, waves, and wind. Assimilation methods (Peytavin et al. 2021) and plastic-specific transport models have also been investigated (Sainte-Rose et al. 2022). On the sensing front, satellite imaging (Park et al. 2021, 2022) and remote sensing techniques (de Vries et al. 2021) are being developed to acquire field data. These models provide a picture of where plastics are located and how they move within this region, hence creating a dynamic view of present and future plastic density (similar to Figure 1, yet evolving over time). Our objective is to integrate these predictions directly into an optimal routing problem, so the system naturally accounts for plastic movements, which is crucial because both the system and the plastics are moving at comparable speeds. A central challenge is to account for the fact that the collection process removes plastic from the oceans and, as such, should directly impact the (estimate of) future plastic density. Hence, plastic density cannot be seen as an exogenous input to our model only, it is also impacted by our routing decisions.

To summarize, our objective is to find a route for the system and to schedule extractions of the retention zone, in order to maximize the total quantity of plastic collected by the system. In particular, we need to account for weather and operational constraints, plastic dynamics, and the direct impact of our decision on future plastic density.

1.2. Contributions and structure

In this work, we develop and validate an optimization approach to jointly optimize the routing and the extractions of the plastic collection system. After reviewing the relevant literature in Section 2, we make the following contributions:

- In Section 3, by discretizing space and time, we model the routing and scheduling decisions as paths in a directed acyclic graph (DAG). Among others, this model can account for relevant operational and weather constraints and provides efficient dynamic programming algorithms for longest-path type of optimization problems.

- Under this lens, the quantity of plastic collected can be seen as edge length in this graph and our problem as a longest path optimization problem. However, due to the direct impact of our routing decisions on future plastic density, our resulting optimization problem is a non-linear and non-decomposable longest path problem. In Section 4, we formally analyze the structure of our path-dependent edge lengths, which resembles structure arising in covering problems. We derive lower and upper bounds on the estimation error obtained when ignoring the path dependency, which will serve as the basis for our algorithmic strategy.
- We propose a search-and-bound strategy to efficiently find a high-quality solution for this class of problems, with certificates of near-optimality (Section 4). Our algorithm leverages a linear relaxation of the problem solvable via dynamic programming to efficiently search through the space of trajectories by combining geographical clustering with terminal values of the DP approach. We also propose a tailored branch-and-bound scheme to solve this class of problems exactly, using our search-and-bound algorithm as the root node analysis. On small instances (up to 3-day planning), our search-and-bound strategy finds an optimal solution, while scaling better with respect to the problem size than exact approaches.
- Finally, we evaluate the benefit of our search-and-bound algorithm on a one-year dataset of ocean weather conditions and plastic density in Section 5. We find that our optimization approach yields at least a 60% improvement in terms of average collection efficiency compared with their current routing strategy. In particular, we observe greater benefits (+100%) during winter months, because weather conditions (and wave height in particular) are limiting the ability to extract, hence exacerbating the benefit of jointly optimizing the route and the extraction schedule. In addition, our algorithm allows The Ocean Cleanup to explore the non-linear impact of strategic system dimensioning decisions (e.g., span of the system and size of the retention zone).

2. Literature Review

Our problem can be summarized as a ship joint routing and scheduling problem, where the objective is to steer the system in the GPGP and schedule extractions (i.e., emptying of the retention zone) in order to maximize the quantity of plastic collected. In Section 2.1, we review the optimization literature related to marine operations and ship routing. We then focus on methods for fishing optimization, which is similar to our plastic collection problem. Eventually, we will model our problem as that of a longest path in an appropriately defined graph, so we review the literature on longest path optimization in Section 2.3.

2.1. Optimization for ship routing problems

Following Granado et al. (2021), we divide the literature into weather and tactical routing.

In weather routing, the objective is to find a route that connects a given origin with a given destination and minimizes travel time or fuel consumption, which depend on weather and navigation conditions. The typical planning horizon in weather routing is a few weeks. The great circle passing through these two locations provides the shortest route in terms of travel distance. So, an optimal route is often to be found in the vicinity of the shortest route. Most approaches create a discrete grid of potential locations around the great circle using isochrone lines (James 1957, Hagiwara and Spaans 1987) or a fixed grid (Zoppoli 1972, de Wit 1990). By representing a trajectory as a sequence of locations, the weather routing problem can thus be formulated as a shortest path problem, which can be solved efficiently by Dynamic Programming (DP; Zoppoli 1972, de Wit 1990, Meng and Wang 2011, Ting and Tzeng 2003, Aydin et al. 2017) or Dijkstra's algorithm (Takashima et al. 2009, Skoglund 2012, Sen and Padhy 2015). Additional decision variables, such as engine power in Shao et al. (2012), can be modeled within a shortest path formulation by extending the description of the 'state' of the ship. Heuristic methods have also been used to deal with more complex objectives or constraints, such as simulated annealing (Kosmas and Vlachos 2012), the A* algorithm (Yoon et al. 2018, Langbein et al. 2011), or particle swarm optimization methods (Zheng et al. 2019). Recently, Cheng and Zhang (2018), Chen et al. (2019) used reinforcement learning approach to optimize the route while learning the complex dynamics between waves and speed or fuel consumption. We refer to Zis et al. (2020) for a comprehensive review on ship weather routing.

For our problem, we adopt a similar modeling paradigm by discretizing the location of the system in the GPGP. However, we adopt a more fine-grained discretization of time (3-hour time steps) and space (8 km), and extend the state variable to account for extraction decisions as well, so our resulting graph is of much larger scale, i.e., in the order of 10^6 nodes. In addition, the destination in our problem is not fixed, which can lead to more complex trajectories, such as circling or crossing. In terms of objective, we assume that the fuel efficiency does not depend on the routing decision because of the limited propelling speed so our primary objective is to maximize the amount of plastic collected in a given amount of time. After appropriately defining edge weights, we formulate our problem as a longest path optimization problem and solve it using DP strategies similar to the ones used in weather ship routing.

Tactical ship routing consists in finding the lowest cost route for a ship that needs to visit different locations (e.g., a cargo ship visiting different ports). Since the time horizon is long (several weeks or months) and the ports are fixed isolated location, the problem can be formulated as a Traveling Salesperson Problem (TSP), solved by branch-and-bound (Appelgren 1971, Stalhane et al. 2015), branch-cut-and-price (Battarra et al. 2014), heuristic methods (Malaguti et al. 2018), or DP (Fagerholt and Christiansen 2000). We refer to Christiansen et al. (2004) for a comprehensive review of the literature and its connection to supply chain management.

2.2. Fish routing

Among all maritime activities, fishing is the most comparable to our plastic collection problem because the objective is to capture floating elements in the oceans.

Before solving any route optimization problem, one needs to first predict the density of fish at different locations. However, unlike plastic, fishes are actively moving, which makes their precise location highly unpredictable. Instead, most works describe fish density with coarse granular distributions (see Robinson et al. 2017, for a review). Unfortunately, estimating the accuracy of these different approaches remains an open challenge. Indeed, 94% of the studies reviewed by Robinson et al. (2017) failed to report the uncertainty of their model.

In the weather routing literature, algorithms that consider wave and wind forecast to design safe and efficient routes have been applied to fishing (e.g., Vettor et al. 2016). In these use cases, the objective of maximizing the quantity of fish collected is captured in the choice of the target destination, and is typically left to the end-user. This implementation bypasses the issue of inaccurate predictions by letting the human user identify (based on quantitative models and intuition) the destination. To the best of our knowledge, no weather ship routing approach uses quantitative fish density predictions directly as an input to optimize the short-term (within the next days) route of fishing ships. Instead, predictions on the presence and movements of fish banks or 'clusters' are mostly used as locations in a tactical ship routing problem. For tuna fishing for example, floating devices are dispersed in the ocean to attract fishes. Groba et al. (2015, 2018, 2020) model the problem of visiting all devices as a dynamic TSP, where locations can drift due to sea current.

2.3. Optimization for longest path

Given weights on the edges of a graph, the length of a path is defined as the sum of the weights of the edges composing the path. The problem of finding the longest path in a graph is shown to be $\mathcal{NP}$-complete as a generalization of the Hamiltonian path problem (Karp 2010). Actually, the longest path problem cannot be approximated in polynomial time unless $\mathcal{P} = \mathcal{NP}$, as proved by Karger et al. (1997) for undirected and Björklund et al. (2004) for directed graph.

In contrast, finding the shortest path in a graph can be solved in polynomial time using algorithms like the greedy-type Dijkstra's algorithm (Dantzig 1960, Dijkstra 1959) or the Bellman–Ford algorithm (Shimbel 1954, Ford 1956, Bellman 1958, Moore 1959). We refer to Pollack and Wiebenson (1960), Schrijver (2012) for comprehensive reviews. Understanding the structural differences between the longest and shortest path problems and their implications for problem complexity has been a vivid research topic (see, e.g., Cormen et al. 2022), unravelling conceptual connections between shortest path algorithms and DP (Sniedovich 2006).

Nonetheless, polynomial time algorithms for longest path problems exist for particular classes of graphs such as trees (Bulterman et al. 2002, Uehara and Uno 2007), block graphs, cactus graphs (Uehara and Uno 2007), and cocomparability graphs (Ioannidou and Nikolopoulos 2013). The graph we propose in Section 3.1 is a DAG. The longest path problem in a DAG can be solved in linear time by transforming it into the shortest path problem (Pandit 1962, Cormen et al. 2022), or using DP on the topological sort of the DAG (Madraki and Judd 2019). The DAG in our project has a natural topological sort, and we use DP in Section 3.2 to solve linear longest path problems.

3. Graph-Based Routing Model

In this section, we propose a graph-based formulation for our problem. By discretizing time and space, we show in Section 3.1 how the routing decision can be modeled as a path in a sparse directed acyclic graph. Accordingly, longest path optimization problems can be solved efficiently over this graph using DP, as presented in Section 3.2. We conclude this section by discussing how extraction scheduling decisions can be incorporated as well—details are deferred to Section A.3 of the Electronic Companion (EC)—and identifying a set of tractable optimization problems we can solve in Section 3.3.

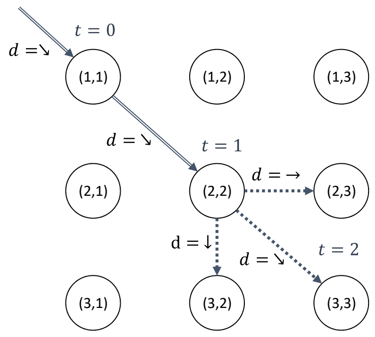
3.1. Discretization and graph representation

To describe the system's trajectory and model the key decisions and constraints of our problem, we discretize space onto a finite grid, as represented in Figure 3. Similarly, we divide our planning horizon using a fixed timestep. In our implementation, we use an 8-km step to discretize space and a 3-hour time step. Denoting $\mathcal{L}$ the set of all possible locations and $\mathcal{T} := \{0, 1, \ldots, T\}$ the set of time periods, we can represent a system trajectory as a sequence of locations, $\{\ell_t\}_{t\in\mathcal{T}}$ with $\ell_t \in \mathcal{L}$.

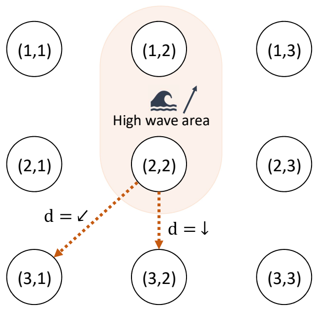
However, as explained in Section 1.1, the steering direction also plays an important role in our problem because of operational (e.g., no sharp turn) and weather constraints (e.g., if the wave height exceeds 4.5 meters, need to navigate against the waves). Hence, at a given time t, knowledge of the current location ℓ_t is not sufficient to determine whether the constraints are satisfied and what the accessible next locations are. Accordingly, we describe a trajectory by a sequence $\{(\ell_t, d_t)\}_{t\in\mathcal{T}}$, where ℓ_t is the location at time t and $d_t \in \mathcal{D}$ is the steering direction at time t. Here, $\mathcal{D}$ denotes the (finite) set of allowable steering directions in our grid, $\mathcal{D} := \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow\}$. For example, we can easily enforce the no-sharp-turn constraint, by limiting the change in direction between t and $t+1$, i.e., the angle between d_t and d_{t+1} (no more than 45° in our case).

Figure 3a shows an example of a discretized trajectory. For simplicity, we associate $\mathcal{L}$ with a set of discrete coordinates in $\mathbb{R}^2$, which are in a one-to-one correspondence with the latitude and longitude of the system. In this example, at $t=0$, the system is at location $(0,0)$, moving south east ($\searrow$). It keeps the same steering direction at $t=1$ and reaches $(1,1)$. At $t=2$, it can reach three different locations depending on whether it continues south east ($d_2 = \searrow$) or decides to change course and move $\rightarrow$ or $\downarrow$. Because the propelling speed of our system is limited (in order not to catch any marine life), we use discretization steps for time (3 hours) and space (8 km) that are consistent with this low propelling speed (1–1.5 knots) and assume that the system in one location can only reach the neighbouring locations at the next time period. We could relax this assumption and adopt a finer discretization strategy to account for travel time differences between diagonal and horizontal/vertical moves or allow for different propelling speed depending on the steering direction (e.g., to maintain a constant speed relative to water).

Using terminology from DP, we refer to the triplet $s := (\ell, d, t) \in \mathcal{L} \times \mathcal{D} \in \mathcal{T}$ as the state of the system. For each state $s = (\ell, d, t)$, we can then define its set of *successors*, i.e., the set of admissible next states $s' = (\ell', d', t+1)$ that satisfy all operational and weather constraints, such as:



(a) Discretization of path



(b) High wave region and direction at $t=1$

Figure 3 Example: routing in a small grid.

- **Consistency between locations and directions:** The next location ℓ' needs to correspond to the location reached from ℓ after following the direction d', which we could express algebraically as "$d' = \ell' - \ell$" after appropriately mapping $\mathcal{L}$ and $\mathcal{D}$ to vectors in $\mathbb{R}^2$.
- **No sharp angles:** In our problem, the angle between the steering directions d and d' is at most 45 degrees (or $\pi/4$). For example, for $s = ((2,2), \searrow, 1)$ in Figure 3a, we must have $d' \in \{\rightarrow, \searrow, \downarrow\}$.
- **High-wave regions:** When the wave height exceeds 4.5 meters, the system has to navigate against the waves. For example, in Figure 3a-3b, assume that the location at $t=1$ is in a high-wave region with waves going north-north-east. Then, according to this constraint, the next direction d' can only be $\swarrow$ or $\downarrow$.

Our model can thus account for any constraint defined on the location or direction of the system—we provide a list of the operational constraints of our problem in EC Section A.1. In Section A.3, we describe how to extend the state space further, to incorporate the decision of extraction scheduling and impose the associated constraints (in particular, extraction can only be performed when the wave height is below 2.5 meters). All together, these constraints define the successors of a state s, which we concisely denote succ(s). For the example, in Figure 3, we have succ$((2,2), \searrow, 1) = \{((3,2), \downarrow, 2)\}$. In other words, the system only has one feasible next state given the current state and weather conditions.

With these notations, we can represent admissible trajectories as paths on a graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$. The set of nodes $\mathcal{S}$ can be naturally partitioned by the time period $t \in \mathcal{T}$, i.e., $\mathcal{S} = \cup_{t \in \mathcal{T}} \mathcal{S}_t$, where $\mathcal{S}_t$ is the set of feasible states at time t and is defined recursively. At time $t=0$, if we are given an initial location ℓ_0 only, then the set of all possible initial states is $\mathcal{S}_0 = \{(\ell_0, d, 0) : d \in \mathcal{D}\}$. We then apply the recursion $\mathcal{S}_{t+1} = \cup_{s \in \mathcal{S}_t}$ succ(s). This offers the flexibility to fix (or not) the starting/ending point of the trajectory. Similarly, the set of edges can be decomposed into $\mathcal{E} = \cup_{t=1}^{T}\mathcal{E}_t$, with $\mathcal{E}_{t+1} = \{(s,s') \in \mathcal{S}_t \times \mathcal{S}_{t+1} \,:\, s' \in \mathrm{succ}(s)\}$. In particular, observe that the graph $\mathcal{G}$ is a DAG. Furthermore, it is relatively sparse. Because of the no-sharp-turn constraint, the number of edges satisfies $|\mathcal{E}_t| = \mathcal{O}(|\mathcal{S}_t|)$. Figure A.1 in the EC shows the graph corresponding to the example of Figure 3. In our implementation, we plan for 7 days with 3-hour time steps, so $T = 7 \times 8 = 56$ and the grid of all reachable locations is of size $|\mathcal{L}| = (56+1+56) \times (56+1+56) = 12,769$. Hence, for each time t, the number of possible states for time t is bounded as follows: $|\mathcal{S}_t| \leq |\mathcal{S}| \times |\mathcal{D}| \approx 10^5$.

3.2. Efficient search for longest path

Thanks to the convenient structure of the graph $\mathcal{G}$, given fixed weights on the edges, w_e for $e \in \mathcal{E}$, we can efficiently find the longest path, i.e., the reward-maximizing trajectory. Assuming that w_e is associated with the quantity of plastic collected when the system moves along edge e at time t, our plastic collection problem would be equivalent to finding the longest path in $\mathcal{G}$ with edges weighted by $\boldsymbol{w}$. Because $\mathcal{G}$ is a DAG, the longest path can be found using a DP algorithm. We present the key ingredients and intuition in this section for the sake of completeness and defer a complete description of the longest path algorithm (Algorithm 1) in Appendix.

For any state $s \in \mathcal{S}_{t+1}$, let $V^{t+1}(s)$ denote the length of the longest path connecting s to $\mathcal{S}_0$:

$$V^{t+1}(s) := \max_{s_0 \in \mathcal{S}_0, \ldots, s_t \in \mathcal{S}_t} \sum_{\tau=0}^{t} w_{s_\tau, s_{\tau+1}} \text{ with } s_{t+1} = s.$$

The key idea in the DP algorithm is that a solution of the optimization problem above can be computed recursively by connecting the longest path between $\mathcal{S}_0$ and s' and the edge (s', s) for some $s' \in \mathcal{S}_t$, i.e., $V^{t+1}(s) = \max_{s' \in \mathcal{S}_t : s \in \mathrm{succ}(s')} \left\{ w_{s',s} + V^t(s') \right\}$. The latter maximization problem is solved by exhaustively searching through $\mathcal{S}_t$.

Algorithm 1 proceeds by recursively computing the values $V^t(s)$ for $t = 0, \ldots, T$. At the end, Algorithm 1 returns, for every possible terminal state $s \in \mathcal{S}_T$, the value of the longest path problem across all possible paths terminating at s, $V^T(s)$, alongside a candidate path achieving this value. Figure 4 shows an example of a 7-day collection route obtained by applying Algorithm 1.

3.3. Summary: Power of the graph-based modeling

In this section, we propose a graph-based model to represent the routing decision as a path in a sparse DAG, $\mathcal{G}$. In EC Section A.3, we show how we can extend the state space $\mathcal{S}$ of our system

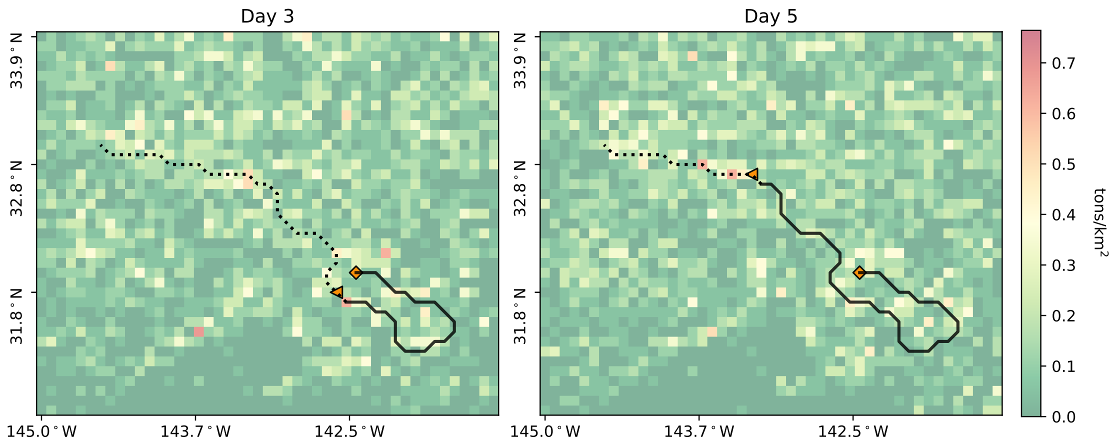


Figure 4 Example of an optimal 7-day route (starting at the diamond location on day 1), represented on day 3 and day 5 (triangle locations). The colors correspond to plastic density for each day.

and the graph $\mathcal{G}$ to also account for extraction scheduling. Hence, we obtain a similar DAG where each path now corresponds to a sequence of routing and extraction scheduling decisions.

The DP algorithm described in Section 3.2 (Algorithm 1) is an efficient approach for solving longest path problems over this graph. In this work, we will be particularly interested in problems where rewards are associated with states instead of edges, i.e., longest path problems of the form

$$\max_{\boldsymbol{x} \in \mathcal{X}} \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}_t} r_s^t x_s^t, \tag{1}$$

where $x_s^t \in \{0,1\}$ indicates whether the system is in state s at time t and $\mathcal{X}$ denotes the set of admissible such binary variables. A formal definition of the feasible set $\mathcal{X}$ is provided in A.4. Of course, problems of the form (1) can be solved by Algorithm 1, as longest path problems with weights $w_{s,s'} = r_{s'}^t$ for $(s,s') \in \mathcal{E}_t$. Conceptually, one can interpret the DP approach as an efficient partitioning of the set $\mathcal{X}$. At the end of Algorithm 1, the set of trajectories $\mathcal{X}$ is partitioned according to the terminal state they reach. $V^T(s)$ corresponds to the value of the longest path problem across all paths terminating at s, and we can get one path achieving this value, denoted $\boldsymbol{x}^s$.

4. Path-Dependent Reward Structure and Algorithms

Since our graph-based formulation presented in Section 3.1 allows us to efficiently find trajectories that maximize a given reward, a natural approach is to cast our plastic collection problem as a longest path problem. Because plastic are freely floating in the oceans, however, plastic particles are constantly moving and the plastic-collecting system directly impacts these dynamics. We present the dynamics of plastic movements in the absence of any collection in Section 4.1, and then derive the resulting dynamics for our objective function in Section 4.2. In particular, the rewards (or edge length) we need to consider are path-dependent, i.e., they depend on the entire past trajectory of the system. We analyze the structure of such path-dependent rewards in Section 4.3. Based on this analysis, we propose an efficient search-and-bound algorithm to find near-optimal solutions to the resulting non-linear optimization problem efficiently (Section 4.4), and a tailored branch-and-bound scheme that solves it to optimality (Section 4.5). We evaluate our algorithms numerically in Section 4.6.

4.1. Fluid mechanics model of free-floating plastic dispersal

Plastic particles move passively in the oceans and, as such, their movements can be modeled and predicted using fluid dispersal models. Among others, the engineering team at The Ocean Cleanup models the velocity of plastic particles in the oceans using data on sea currents and waves, and taking into account the Stokes drift (Stokes 1847) and Eddy diffusivity (Taylor 1915) phenomena.

Denoting r_ℓ^t the quantity of plastic present at time $t \in \mathcal{T}$ and at location $\ell \in \mathcal{L}$, these fluid dispersal models provide us with estimates on the quantity of plastic present in the region at times $(\boldsymbol{r}^0, \dots, \boldsymbol{r}^T)$ as well as structural relationships connecting the vectors. Formally, from the models developed by The Ocean Cleanup, we also obtain matrices $\boldsymbol{Q}^t \in \mathbb{R}_+^{\mathcal{L}\times\mathcal{L}}$ such that

$$\boldsymbol{r}^{t+1} = \boldsymbol{Q}^t \boldsymbol{r}^t. \tag{2}$$

Each entry $Q_{\ell,\ell'}^t$ of the matrix $\boldsymbol{Q}^t$ indicates the fraction of plastic present at location ℓ' at time t that moves to location ℓ at time $t+1$. If the total quantity of plastic is constant (which is a reasonable assumption given our relatively short planning horizon), then the matrix $\boldsymbol{Q}^t$ should be left-stochastic (i.e., $\sum_{\ell\in\mathcal{L}} Q_{\ell',\ell}^t = 1$ for all $\ell \in \mathcal{L}$). In this case, we could interpret $\boldsymbol{Q}^t$ as the transition matrix of a Markov process. In our approach, we only require that $\boldsymbol{Q}^t$ has non-negative entries.

Remark 1. In practice, the matrices $\boldsymbol{Q}^t$ are large, $125,000 \times 125,000$ in our implementation, so matrix-vector products involving $\boldsymbol{Q}^t$'s can be computationally challenging. Actually, the construction of the matrices $\boldsymbol{Q}^t$ from the particle-level fluid dispersal model is the most time-consuming step. We discuss computational aspects of the plastic dynamics model in Section C of the EC.

4.2. Path dependency

Given this information, we now define a relevant objective for our longest path problem. While the plastic density vectors (or 'maps') $\boldsymbol{r}^t$, $t \in \mathcal{T}$, defined in the previous section, describe the plastic dynamics in absence of any collection process, our system actively removes plastic from the ocean and the collected plastic no longer evolves according to (2). In other words, the quantity of plastic collected by our system (and the locations where this plastic has been collected) directly impacts the future spatial distribution of plastic. It is relevant to our optimization problem because our system moves at a speed comparable to that of the plastic. We refer to this phenomenon as *path dependency* and now appropriately define a reward (or length) vector for our optimization problem that takes this phenomenon into account.

Let us denote the location of the system at time t through a one-hot vector $\boldsymbol{x}^t \in \{0,1\}^{\mathcal{L}}$, where $x^t_\ell = 1$ if and only if the system is in ℓ at time t. We denote by $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} \in \mathbb{R}^{\mathcal{L}}_+$ the quantity of plastic present (or reward) associated with each location at time t, where $\boldsymbol{x}^{0:t-1}$ concisely denotes the sequence $\{\boldsymbol{x}^0, \dots, \boldsymbol{x}^{t-1}\}$ and emphasizes the dependency on the past trajectory. If the system is in location ℓ at time t, it collects a fraction $\alpha \in [0,1]$ of the plastic present. Hence, it collects $\alpha r^t_{\ell|\boldsymbol{x}^{0:t-1}}$ and the remaining $(1-\alpha) r^t_{\ell|\boldsymbol{x}^{0:t-1}}$ continues to float in the ocean, together with the plastic present in other locations, $r^t_{\ell'|\boldsymbol{x}^{0:t-1}}$ for $\ell' \neq \ell$. All together, the spatial density of plastic at time $t+1$, $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$, should depend on $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ and $\boldsymbol{x}^t$ through the following recursion

$$\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} = \boldsymbol{Q}^t \left(\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} - \alpha\, \boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} \circ \boldsymbol{x}^t \right), \tag{3}$$

where $\circ$ denotes the Hadamard or element-wise product between two vectors. Hence, $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} - \alpha\, \boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} \circ \boldsymbol{x}^t$ corresponds to the density map where we remove a fraction α of the plastic in the location of the cleaning system.[1] Note that, since $\boldsymbol{Q}^t$, $\boldsymbol{x}^t$, and $\boldsymbol{r}^0$ have non-negative entries, one can show by induction that $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \geq 0$.

With these dynamics in mind, the problem of jointly routing the system and scheduling the extractions in order to collect the maximum amount of plastic possible can be formulated as the following longest path optimization problem:

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}_t} r^t_{s|\boldsymbol{x}^{0:t-1}} x^t_s \quad \text{s.t.} \quad \boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} = \mathcal{Q}^t(\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}, \boldsymbol{x}^t), \tag{4}$$

which is analogous to the longest path problem (1) except that the rewards are no longer fixed but also depend on the past decisions, $\boldsymbol{x}^{0:t}$. Unfortunately, Problem (4) is much more challenging to solve than (1) because the objective is non-linear. To better understand the dynamics and complexities of the problem, we study analytically the path-dependent reward vectors defined by the recursion (3) in the following section.

Remark 2. Note that, with a slight abuse of notations, we use the variable x_s^t in Problem (4) to encode for the *state* of the system at time t, while plastic dynamics (3) are described using binary variables x_ℓ^t encoding for the *location* of the system (location being one component of the state only)—and similarly for the associated rewards. However, it should be clear that we can recover the location from the system's state via a simple affine mapping and that the reward dynamics described at a location level in (3) imply similar dynamics for the state rewards. We formally define this mapping in the EC, Section B.1, and introduce a generic operator $\mathcal{Q}^t$ in the optimization problem (4) to concisely capture the resulting dynamics on the state variables/rewards. In the remainder of this section, for ease of notations, we will implicitly work with location-based $\boldsymbol{x}$ variables when analyzing the structure of the rewards generated by the recursive formula (3) but refer to the state-level variables when describing optimization problems and algorithms. This simplification is valid because the mapping between the two descriptions is monotonous.

4.3. Reward decomposition

We analyze the structure of the path-dependent reward $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$ to inform our algorithmic strategy.

To build intuition, we start by the special case where plastic does not move, i.e., when the matrices $\boldsymbol{Q}^t$ are the identity matrices. In this case, in the absence of any plastic collection, the plastic density maps $\boldsymbol{r}^t$ defined by Equation (2) are constant over time, $\boldsymbol{r}^0 = \cdots = \boldsymbol{r}^T =: \boldsymbol{r}$. Accordingly, we drop the time superscript, although the path-dependent reward, $\boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}}$, still depends on time t through the past trajectory $\boldsymbol{x}^{0:t-1}$. In this case, we have the following expansion:

Lemma 1. *When the matrices $\boldsymbol{Q}^t$ are all equal to the identity matrix, we have*

$$\boldsymbol{r}_{|\boldsymbol{x}^{0:t}} = \boldsymbol{r} + \sum_{1\le k\le t} (-\alpha)^k \sum_{0\le t_1<\cdots<t_k\le t} \boldsymbol{r}\circ \boldsymbol{x}^{t_1}\circ\cdots\circ \boldsymbol{x}^{t_k}.$$

Proof of Lemma 1 In this case, the plastic dynamics (3) simplifies as

$$\boldsymbol{r}_{|\boldsymbol{x}^{0:t}} = \boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}} - \alpha\, \boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}} \circ \boldsymbol{x}^t = \boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}} \circ (\boldsymbol{1} - \alpha \boldsymbol{x}^t) = \boldsymbol{r}\circ(\boldsymbol{1}-\alpha\boldsymbol{x}^0)\circ\cdots\circ(\boldsymbol{1}-\alpha\boldsymbol{x}^t).$$

The Hadamard product being commutative and associative, we can use the classical polynomial expansion technique to obtain

$$\boldsymbol{r}_{|\boldsymbol{x}^{0:t}} = \boldsymbol{r} \circ \left[\sum_{0 \le k \le t} (-\alpha)^k \sum_{0 \le t_1 < \cdots < t_k \le t} \boldsymbol{x}^{t_1} \circ \cdots \circ \boldsymbol{x}^{t_k} \right] = \sum_{0 \le k \le t} (-\alpha)^k \sum_{0 \le t_1 < \cdots < t_k \le t} \boldsymbol{r} \circ \boldsymbol{x}^{t_1} \circ \cdots \circ \boldsymbol{x}^{t_k}.$$

□

Lemma 1 shows that the path-dependent reward $\boldsymbol{r}_{|\boldsymbol{x}^{0:t}}$ can be computed from $\boldsymbol{r}$ by applying successive corrections. The kth-order correction in this expansion involves Hadamard products of the form $\boldsymbol{x}^{t_1} \circ \cdots \circ \boldsymbol{x}^{t_k}$, each of them being different from the $\mathbf{0}$ vector if and only if there exists a location ℓ such that $x_\ell^{t_1} = \cdots = x^{t_k} = 1$. In other words, the first-order correction consists in removing a fraction α of the plastic in locations visited at least once by the system; the second-order correction adds a fraction α^2 of the plastic in locations visited as least twice by the system; and so on. In the general case, the plastic particles move according to the matrix $\boldsymbol{Q}^t$ so they are not assigned to a fixed location. Still, the intuition of Lemma 1 holds: the path-dependent rewards $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ can be obtained from the original rewards $\boldsymbol{r}^t$ by removing a fraction α of the plastic particles encountered once, adding a fraction α^2 of the plastic particles encountered twice,... We derive analytically the order-two expansion (in α) of the path-dependent reward in the general case in Section B.2 of the EC.

Remark 3. The terms in the expansion in Lemma 1 decay exponentially in k because of the α^k term and because the number of locations being updated decreases:

$$\sum_{0 \le t_1 < \cdots < t_k < t_{k+1} \le t} \boldsymbol{x}^{t_1} \circ \cdots \circ \boldsymbol{x}^{t_k} \circ \boldsymbol{x}^{t_{k+1}} \le \sum_{0 \le t_1 < \cdots < t_k \le t} \boldsymbol{x}^{t_1} \circ \cdots \circ \boldsymbol{x}^{t_k}, \quad \forall k \ge 0.$$

The structure of these expansions is analogous to the inclusion-exclusion principle from probability. From this analogy, one can expect that truncating the expansion at a fixed order k with k even (resp. odd) leads to an upper (resp. lower) bound on the path-dependent reward. Indeed, Proposition 1 shows that zero-th and first-order expansion provide valid upper and lower bound respectively on the path-dependent reward.

Proposition 1. *The path-dependent reward* $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$ *satisfies the following bounds:*

$$\boldsymbol{r}^{t+1} - \alpha \sum_{0 \le t_1 \le t} \left(\boldsymbol{Q}^t \times \cdots \times \boldsymbol{Q}^{t_1}\right)\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right) \quad \le \quad \boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \quad \le \quad \boldsymbol{r}^{t+1}. \tag{5}$$

Proof of Proposition 1 We prove the result by induction. For $t=0$, $\boldsymbol{r}^1_{|\boldsymbol{x}^{0:0}} = \boldsymbol{r}^1 - \alpha \boldsymbol{Q}^0 \left(\boldsymbol{r}^0 \circ \boldsymbol{x}^0\right)$. Hence, in this case, $\boldsymbol{r}^1_{|\boldsymbol{x}^{0:0}}$ is exactly equal to the lower bound in (5). Since $\boldsymbol{Q}^0$, $\boldsymbol{r}^0$, and $\boldsymbol{x}^0$ have non-negative entries, $\boldsymbol{Q}^0 \left(\boldsymbol{r}^0 \circ \boldsymbol{x}^0\right) \geq 0$ and the upper bound in (5) holds as well.

Let us now assume that the bounds (5) hold for some $t \geq 0$ and let us show that they also hold for $t+1$.

For the upper bound,

$$\boldsymbol{r}^{t+2}_{|\boldsymbol{x}^{0:t+1}} = \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} - \alpha \boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \circ \boldsymbol{x}^{t+1}\right) \leq \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \leq \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1} = \boldsymbol{r}^{t+2}.$$

For the lower bound

$$\begin{aligned}
\boldsymbol{r}^{t+2}_{|\boldsymbol{x}^{0:t+1}} &= \boldsymbol{Q}^{t+1}\left[\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \circ \left(\mathbf{1} - \alpha \boldsymbol{x}^{t+1}\right)\right] \\
&\geq \boldsymbol{Q}^{t+1}\left[\left(\boldsymbol{r}^{t+1} - \alpha \sum_{0 \leq t_1 \leq t} \left(\boldsymbol{Q}^t \times \cdots \times \boldsymbol{Q}^{t_1}\right)\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right)\right) \circ \left(\mathbf{1} - \alpha \boldsymbol{x}^{t+1}\right)\right] \\
&= \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1} - \alpha \boldsymbol{Q}^{t+1} \sum_{0 \leq t_1 \leq t} \left(\boldsymbol{Q}^t \times \cdots \times \boldsymbol{Q}^{t_1}\right)\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right) - \alpha \boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1} \circ \boldsymbol{x}^{t+1}\right) + \alpha^2 \underbrace{\cdots}_{\geq 0} \\
&\geq \boldsymbol{r}^{t+2} - \alpha \sum_{0 \leq t_1 \leq t+1} \boldsymbol{Q}^{t+1} \times \left(\boldsymbol{Q}^t \times \cdots \times \boldsymbol{Q}^{t_1}\right)\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right),
\end{aligned}$$

which concludes the proof. $\square$

Our analysis of the structure of the path-dependent rewards highlights the fact that path dependency is essentially a double-counting issue: The path-dependent rewards count the quantity of plastic that can be collected by the system. If a plastic particle has already been captured by the system in the past, it should not be counted again to estimate future rewards. This structure is also present in optimization problems with coverage objectives. In the maximal covering location problem (Church and ReVelle 1974) for example, the total coverage is not the sum of the regions covered by each facility separately because regions covered by more than one facility should not be counted multiple times. Our problem offers a novel real-life example of this classical structure, with the following twists: a large number of 'regions to cover' (plastic particles in the GPGP in our language), dispersion dynamics, and the presence of an efficiency ratio ($\alpha \in (0,1]$).

Conceptually, the reward vectors $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ are order-t polynomials in $\boldsymbol{x}$, leading to the hard non-linear optimization problem (4). However, by Section 3.2, we know how to efficiently maximize linear functions over the feasible set $\mathcal{X}$. Hence, in the rest of this section, we develop and validate the following algorithmic strategy: a linear relaxation of the path-dependent reward and an efficient feasible-solution search to find high-quality solutions quickly, combined with a tailored branching scheme to refine this approximation and ultimately converge to an optimal solution.

REMARK 4. Proposition 1 motivates us to relax (or upper-bound) $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ by a reward vector that does not depend on $\boldsymbol{x}$, and to refine this approximation by branching. Alternatively, we could construct tighter relaxations by using the last k locations ($k \geq 0$), and converge to an optimal solution by taking $k \to T$. Although less scalable than our tailored branch-and-bound, we present and evaluate this alternative strategy in the EC, Section D.5.

4.4. Search-and-bound heuristic

In this section, we propose an algorithm for finding a provable near-optimal solution to Problem (4) quickly. Our algorithm (Algorithm 2) relies on a linear relaxation solvable by DP (Algorithm 1), followed by a search through the feasible space, which is informed by a clustering of the trajectories based on their last location and the terminal values obtained from the DP algorithm, $V^T(s)$. Algorithms 1 and 2 can be interpreted as relaxation-induced searches (Danna et al. 2005) that will constitute the root node analysis in our branch-and-bound scheme.

Instead of solving (4), we solve the linear longest-path problem (1) with the path-independent rewards $\boldsymbol{r}^t$ as the objective. According to Proposition 1, this linear problem provides an upper-bound on (4), i.e., constitutes a valid relaxation. This result is intuitive: ignoring the effect of our collection on future plastic collection leads to an optimistic (i.e., over-) estimate of the plastic we can actually collect. Furthermore, for any t, Proposition 1 states that $\|\boldsymbol{r}^t - \boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}\| = \mathcal{O}(\alpha t)$, hence, the total relaxation error scales at most like αT^2.

For concision, let us omit the time superscript in this section and concisely denote $\boldsymbol{r}$ the plastic density maps obtained by applying the fluid advection equations (2) without any active collection, and $\boldsymbol{r}_{|\boldsymbol{x}}$ its path-dependent version. Both Problem (4) and its linear relaxation optimize over the same feasible space, $\boldsymbol{x} \in \mathcal{X}$, but differ in their objective function, $\langle \boldsymbol{r}_{|\boldsymbol{x}}, \boldsymbol{x} \rangle$ and $\langle \boldsymbol{r}, \boldsymbol{x} \rangle$ respectively, where we use $\langle \cdot, \cdot \rangle$ to denote the inner product between two vectors indexed by time $t \in \mathcal{T}$ and state $s \in \mathcal{S}$. Let us denote $\boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}})$ and $\boldsymbol{x}^\star(\boldsymbol{r})$ their respective solutions.

With these notations, $\boldsymbol{x}^\star(\boldsymbol{r})$ is the solution returned by Algorithm 1. Because $\boldsymbol{r}_{|\boldsymbol{x}} \leq \boldsymbol{r}$ (Proposition 1) and because flows are non-negative,

$$\langle \boldsymbol{r}_{|\boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}})}, \boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}}) \rangle \leq \langle \boldsymbol{r}, \boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}}) \rangle \leq \langle \boldsymbol{r}, \boldsymbol{x}^\star(\boldsymbol{r}) \rangle =: UB,$$

where the last inequality follows from the fact that $\boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}}) \in \mathcal{X}$ is feasible for (1) and the optimality of $\boldsymbol{x}^\star(\boldsymbol{r})$. In addition, $\boldsymbol{x}^\star(\boldsymbol{r}) \in \mathcal{X}$ is feasible for (4) so

$$LB := \langle \boldsymbol{r}_{|\boldsymbol{x}^\star(\boldsymbol{r})}, \boldsymbol{x}^\star(\boldsymbol{r}) \rangle \leq \langle \boldsymbol{r}_{|\boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}})}, \boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}}) \rangle.$$

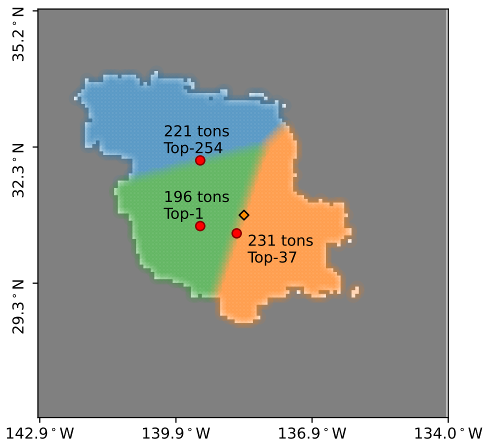


Figure 5 Illustration of our cluster-based search strategy on one 7-day planning instance (Jan 15, 2002). We first exclude locations that are provably suboptimal (grey region), cluster the remaining area into K clusters (here, $K = 3$), and then evaluate one trajectory per cluster (red circle dots). The starting location is indicated by a diamond at the center of the map.

Note that the reward vectors in the two sides of this inequality are different because they correspond to different paths $\boldsymbol{x}^\star(\boldsymbol{r})$ and $\boldsymbol{x}^\star(\boldsymbol{r}_{|\boldsymbol{x}})$. Hence, the path-dependent reward achieved by $\boldsymbol{x}^\star(\boldsymbol{r})$ also provides a valid lower bound on the value of (4). Altogether, Algorithm 1 can be used to return a feasible solution to Problem (4) alongside an optimality gap.

Furthermore, any other feasible solution $\boldsymbol{x} \in \mathcal{X}$ provides a valid, and potentially better, solution. In Algorithm 2, we propose a procedure to search for a better feasible solution $\boldsymbol{x} \in \mathcal{X}$, which we illustrate in Figure 5. Remember that for any terminal state s, Algorithm 1 returns the length (according to $\boldsymbol{r}$) of the longest path reaching s, $V^T(s)$, as well as one path of that length, $\boldsymbol{x}^s$. First, we exclude the terminal states $s \in \mathcal{S}_T$ such that $V^T(s) < LB$, because they cannot contain any solution better than $\boldsymbol{x}^\star(\boldsymbol{r})$ (the outer grey zone in Figure 5). Then, we cluster the space of remaining terminal states into K regions $\mathcal{S}_T^{(k)}$, based on geographical coordinates (we have $K = 3$ in Figure 5). For each region, we consider one candidate trajectory, corresponding to the state *within* $\mathcal{S}_T^{(k)}$ with the largest terminal value $V^T(s)$, and evaluate its path-dependent reward. This step, **($\star$)**, is the most computationally expensive part of Algorithm 2. Alternatively, one could have searched through $\mathcal{S}_T$ by decreasing terminal value $V^T(s)$. As illustrated in Figure 5, however, searching by geographical cluster enforces diversity among the candidate trajectories and lead to better performance. We compare these two search strategies extensively in the EC, Section D.2.

Algorithm 2: Search-and-Bound Algorithm for Problem (4)

```
    Data: Weighted graph $\mathcal{G}$ with dynamic plastic density estimates $\boldsymbol{r}^0$, $\{\boldsymbol{Q}^t\}_{t\in\mathcal{T}}$;
 1  Initialize: Compute $\boldsymbol{r}^t$ for all $t\in\mathcal{T}$ according to (2);
 2  Stage 1: Run Algorithm 1, obtain values $V^T(s)$ and solutions $\boldsymbol{x}^s$ for $s\in\mathcal{S}_T$;
 3  Stage 2: Search-and-bound;
 4  Initialize upper bound $UB=\max_{s\in\mathcal{S}_T} V^T(s)$;
 5  Find $s^\star\in\arg\max_{s\in\mathcal{S}_T} V^T(s)$ and define $\boldsymbol{x}^\star=\boldsymbol{x}^{s^\star}$ ;
 6  Initialize lower bound $LB=\langle \boldsymbol{r}_{|\boldsymbol{x}^\star},\boldsymbol{x}^\star\rangle$, best solution $\hat{\boldsymbol{x}}=\boldsymbol{x}^\star$;
 7  Apply $K$-means clustering to construct the partition $\{s: V^T(s)\geq LB\}=\cup_{k=1}^K \mathcal{S}_T^{(k)}$;
 8  for each region $\mathcal{S}_T^{(k)}$, $k=1,\dots,K$ do
 9      Find $s^{(k)}\in\arg\max_{s\in\mathcal{S}_T^{(k)}} V^T(s)$ and define $\boldsymbol{x}=\boldsymbol{x}^{s^{(k)}}$;
10      $(\star)$ Compute the path-adjusted reward $\boldsymbol{r}_{|\boldsymbol{x}}$;
11      if $\langle \boldsymbol{r}_{|\boldsymbol{x}^s},\boldsymbol{x}^s\rangle > LB$ then
12          Update lower bound $LB=\langle \boldsymbol{r}_{|\boldsymbol{x}^s},\boldsymbol{x}\rangle$;
13          Update best solution $\hat{\boldsymbol{x}}=\boldsymbol{x}$;
14      end
15  end
16  return the solution $\hat{\boldsymbol{x}}$ and optimality gap $(UB-LB)/UB$.
```

Overall, Algorithm 2 uses the value of the relaxed problem (1) as an upper bound on the final value and efficiently search for high-quality feasible solutions to obtain a lower bound, hence the term ‘search-and-bound’.

4.5. Tailored branch-and-bound

In this section, we propose a tailored branch-and-bound algorithm to iteratively refine our upper bound on the value of (4) and converge towards an optimal solution. Recall that in the relaxation, we use the raw density maps $\boldsymbol{r}^t$ (path-*in*dependent) as edge lengths. This relaxation is valid because for any admissible path $\boldsymbol{x}$, $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \leq \boldsymbol{r}^{t+1}$ (Proposition 1). We now show how to refine this approximation when fixing the specific locations visited by the system and use this principle as the basis for our branching scheme.

Let us consider a subset of trajectories $\tilde{\mathcal{X}}\subseteq\mathcal{X}$ such that trajectories within $\tilde{\mathcal{X}}$ must pass through k locations at k particular time points. Formally, we consider an integer k, times $t_1,\dots,t_k\in\mathcal{T}$, and locations $\ell_1, \dots, \ell_k \in \mathcal{L}$ and assume that $\tilde{\mathcal{X}} = \{\boldsymbol{x} \in \mathcal{X} : x_{\ell_i}^{t_i} = 1, \forall i = 1, \dots, k\}$. We construct an upper bound on the path-dependent reward, $\tilde{\boldsymbol{r}}$, as follows:

$$\begin{aligned} \tilde{\boldsymbol{r}}^0 &= \boldsymbol{r}^0, \\ \tilde{\boldsymbol{r}}^{t+1} &= \begin{cases} \boldsymbol{Q}^t \left(\tilde{\boldsymbol{r}}^t - \alpha \tilde{\boldsymbol{r}}^t \circ \boldsymbol{x}^t \right) & \text{if } t \in \{t_1, \dots, t_k\}, \\ \boldsymbol{Q}^t \tilde{\boldsymbol{r}}^t & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

These new reward vectors are indeed tighter upper bounds on $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$:

$$\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \leq \tilde{\boldsymbol{r}}^{t+1} \leq \boldsymbol{r}^{t+1}, \forall t \geq 0, \quad \forall \boldsymbol{x} \in \tilde{\mathcal{X}}. \tag{7}$$

We defer a formal proof of (7) to Section D.3 of the EC. Due to this property, we can relax the path-dependent problem (4) over the restricted feasible space $\tilde{\mathcal{X}}$ into a linear longest-path optimization problem with edge lengths $\tilde{\boldsymbol{r}}$, which is tighter than the original relaxation using $\boldsymbol{r}$. We can apply Algorithm 1 or 2 on this restricted problem to obtain an upper and lower bound. To refine this approximation, we pick a time $t_0 \in \mathcal{T}$ and a location $\ell_0 \in \mathcal{L}$ and partition $\tilde{\mathcal{X}}$ into

$$\begin{tikzcd} & \tilde{\mathcal{X}} \arrow[dl, dash] \arrow[dr, dash] & \\ \tilde{\mathcal{X}}_0 & & \tilde{\mathcal{X}} \setminus \tilde{\mathcal{X}}_0 \end{tikzcd}$$

where $\tilde{\mathcal{X}}_0 := \{\boldsymbol{x} \in \tilde{\mathcal{X}} : x_{\ell_0}^{t_0} = 1\}$. In other words, $\tilde{\mathcal{X}}_0$ fixes a new time/location for the admissible trajectories. Hence, we can construct a tighter upper approximation of the path-dependent rewards over $\tilde{\mathcal{X}}_0$ (the left child node) by applying (6) again. The subproblem $\tilde{\mathcal{X}}_0$ benefits from both a tighter approximation and a reduced search space, so we should expect to effectively reduce the optimality gap on this child node. For the right child node, however, the benefit only comes from reducing the search space from $\tilde{\mathcal{X}}$ to $\tilde{\mathcal{X}} \setminus \tilde{\mathcal{X}}_0$. Because of this imbalance, we expect this branching scheme to experience slow convergence towards an optimal solution, especially for large instances. Nonetheless, theoretically, this branching strategy eventually enumerates all possible trajectories so is guaranteed to converge to an optimal solution after a finite yet exponential number of iterations. We evaluate its numerical behavior, alongside Algorithm 2 in the following section. Details about the implementation of our branch-and-bound are provided in the EC Section D.3.

4.6. Numerical validation

In this section, we evaluate the numerical performance of Algorithm 1 (Alg 1), Algorithm 2 (Alg 2), and tailored branch-and-bound (B&B). For Algorithm 2, we use $K = 12$ clusters. For the tailored branch-and-bound, we impose a limit on the maximum number of branches (50 or 150). We impose no gap nor time limit.

Table 1 Average performance of Algorithm 1, Algorithm 2, and our tailored branch-and-bound on 250 7-day planning instances. We impose a limit on the maximum numbers of branches of 50 for tailored B&B.

	Algorithm 1	Algorithm 2	Tailored B&B
Best solution (LB)	51.378	52.763	52.830
Best bound (UB)	56.236	56.236	54.475
Gap (with respective UB)	9.172%	6.323%	2.918%
Gap (with B&B UB)	5.758%	3.016%	2.918%
Time	14.15 sec	45.60 sec	951.52 sec

First, we consider 250 instances of our 7-day routing problem (see Section 5.1 for a description of the weather and plastic ocean data), so $T = 8 \times 7 = 56$. We generate the instances by considering 50 different starting times (the beginning of each week, except for the first and last weeks of the year) and 5 different initial locations. Table 1 reports the average lower bound, upper bound, optimality gap, and computational time achieved by each method on these instances.

Comparing the performance of Algorithm 1 and Algorithm 2, we observe that Algorithm 2 improves the quality of the best solution, from 51.378 up to 52.763 (+2.7%), the returned upper bound being the same (as expected). This improvement translates into a reduction in the optimality gap returned by the algorithms from 9.2% (Algorithm 1) down to 6.3% for Algorithm 2—a 2.9 percentage points or 32% improvement. The distribution of these gaps (see boxplots in Figure D.4) also shows that Algorithm 2 displays a lower variability in performance across instances, which is desirable. The additional computational time required (45 seconds instead of 14) is noticeable, yet affordable for 7-day planning problems.

Our tailored branch-and-bound algorithm, on the other hand, provides a much narrower improvement in terms of solution quality (+0.1%), and requires 15 minutes on average. Note that, at this scale, it terminates because of the limit on the maximum number of branches (50). For this reason, we do not view our tailored branch-and-bound algorithm as a practical alternative to Algorithm 2 for finding high-quality solutions—Algorithm 2 achieves the same quality and is 20 times faster. However, it returns a significantly tighter lower bound (54.475 on average, −3%). This suggests that the primary benefit of branching is to certify optimality. In particular, we can use this refined upper bound to tighten (and, in our case, halve) the sub-optimality gap of the solutions obtained by Algorithm 1 and 2, which suggests that the solution returned by Algorithm 2 could be much closer to optimality than indicated by the linear relaxation.

We investigate the validity of these observations as the length of the planning horizon, T, varies. Figure 6 reports the average computational time of each method for 6 instances and 14 different

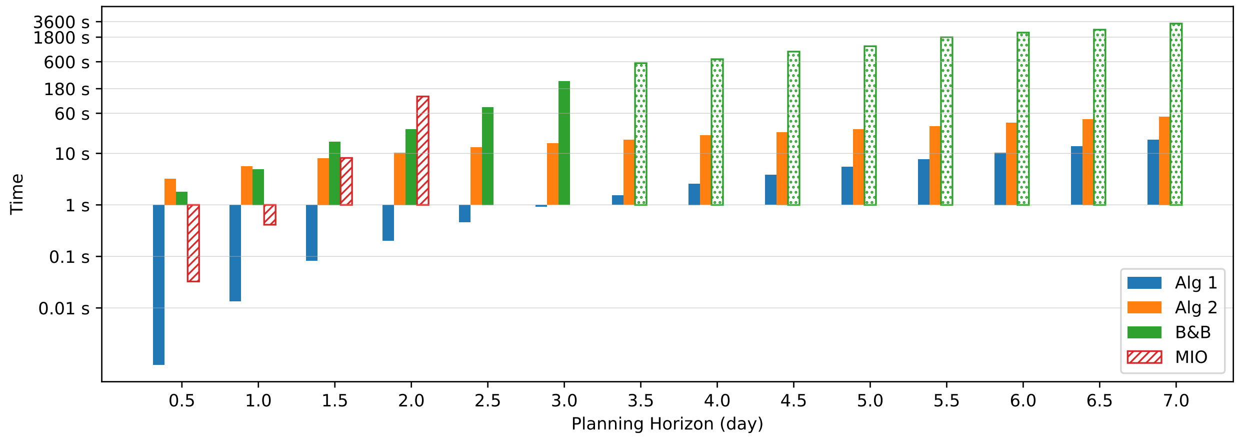


Figure 6 Average computational time (log scale) returned by Algorithm 1, Algorithm 2, and our tailored branch-and-bound algorithm. We also report the time required by an off-the-shelf commercial MIO solver (MIO), excluding pre-computation time required for creating the matrics $\boldsymbol{Q}^t$'s (hashed bars). For B&B, the dotted bars indicate instances where the maximum number of branches (150) triggered algorithm termination.

values of T. Among others, we observe that the relative difference in computational time between Algorithm 1 (blue) and Algorithm 2 (orange) decreases with T, illustrating that the additional search stage does come at an exponential (in T) cost. Second, our tailored branch-and-bound algorithm solves to optimality instances with up to 3-day horizons (beyond that, it terminates because of the limit on the number of branching). Nonetheless, it constitutes a significant improvement over solving a mixed-integer optimization (MIO) formulation of Problem (4) (see Section D.1 of the EC) with commercial solvers (Gurobi in our implementation): For 2-day instances, MIO requires around 4 hours to create all the matrices $\boldsymbol{Q}^t$ (excluded from the time reported in Figure 6) and a couple minutes for solving it, while our B&B does not need such pre-processing (whose time increases exponentially with T) and terminates in less than a minute, a two-order-of-magnitude improvement.

Similarly, Figure D.6 in Section D.4 of the EC reports the average solution quality and gap for each method as the planning horizon increases. Among others, we observe that the solution returned by Algorithm 2 is actually optimal for instances with a planning horizon of less than 3 days (although the returned gap is strictly positive), and that the returned gap over-estimates the true sub-optimality of the solution by at least a factor of two on the larger instances.

5. Numerical Experiment

In this section, we evaluate our method on one-year weather and plastic density data. After presenting our experimental setting in Section 5.1, we compare the performance of different implementations of our algorithm in Section 5.2. In Section 5.3, we delve deeper into the differences in plastic collection efficiency across seasons. Finally, we use our algorithm to investigate the non-linear relationships between some system design decisions (such as total span or size of the retention zone) on the overall efficiency in Section 5.4.

5.1. Experimental setting and implementation choices

We work with one year of weather and plastic density data (year 2002 in our data set). The weather data provides the height and direction of the waves and the wind. The plastic density data is provided as trajectories of a particle-based dispersal model, as described in EC Section C. Consistent with a span of 1.8 km, we consider a plastic collection rate $\alpha = 20\%$. The capacity of the retention zone is fixed to 25 metric tons. We assume that an extraction takes one day (8 time periods).

We divide the year into 13 non-overlapping 28-day periods[2], which we later refer to as 'simulations'. For each simulation, we assume the system starts in the center of the GPGP, whose coordinates are (31.92°N, 142.4°W). We do not impose any restriction on the system's location at the end of each simulation (no depot location).

As a benchmark, we use The Ocean Cleanup's current heuristic, which we formally describe in Section E.1. In short, the benchmark picks the steering direction d leading to the highest distance-weighted reward over the next T time periods. The benchmark does not include any rule for extraction scheduling so we start an extraction as soon as the system reaches capacity.

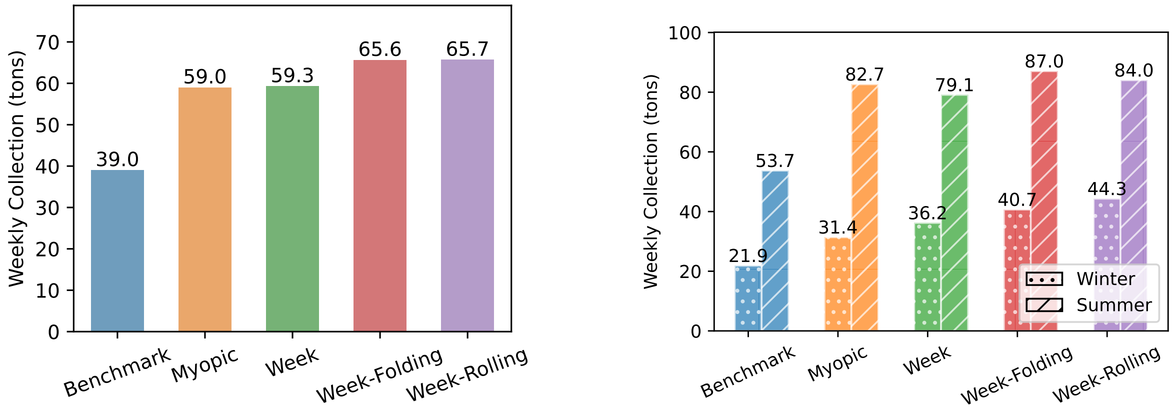
We evaluate the performance of four different implementations of Algorithm 2, with different optimization and implementation horizons (results for Algorithm 1 are presented in the EC, Section E.2): First, we solve our longest path problem (4) for $T = 8$ time periods only (one day), solving and implementing the resulting solution every day. We refer to this implementation as **Myopic**. To be more forward looking, we consider using $T = 56$ time periods instead (one week): At the beginning of each week, we run Algorithm 2, obtain a solution, and implement it for the following 7 days (**Week**). In these two variants, the planning horizon (used in the definition of the optimization problem) and the implementation period are the same. However, they do not have to be. For example, we can use a rolling horizon by solving each day a 7-day longest path problem (4) and implementing its solution for the first day (**Week-Rolling**). Alternatively, we can use a folding (or shrinking) horizon by finding the longest path 'until the end of the week' (i.e., over 7 days in the beginning, 6 days after the first day,...) and implementing its solution for the first day (**Week-Folding**).

We should acknowledge that we choose a 7-day planning horizon as an illustration, because it is simple, tractable, and captures most of the long-term dynamics of our system. However, experiments with varying planning horizon lengths (see Section E.4) suggest that it could be reduced to 5 days without much performance loss. Generally, we recommend a planning horizon long enough to include the next extraction, especially in winter (see discussion in Section 5.3). A longer planning horizon, on the other hand, would be unrealistic in our view, because weather forecasts are not reliable beyond 4/5 days. Unfortunately, our data does not allow us to quantify more precisely the impact of forecast accuracy on performance at this stage. Resolving—every day for the Myopic, Week-Rolling and Week-Folding implementations, and every week for Week—allows us to update our plastic density estimates based on the past-day trajectory, and potentially based on new weather forecasts. Hence, it can help mitigate the path-dependency issues described in Section 4, and improve robustness to forecasting errors in practice.

5.2. Overall improvement in plastic collection rate

Figure 7a represents the weekly quantity of plastic collected by each method, averaged over our 13 four-week simulations. First, Figure 7a illustrates the edge of optimization, with all methods significantly improving over the benchmark—as identified formally by paired t-test with $\leq 10^{-5}$ p-values in Table E.1. Among all methods, Week-Rolling collects the most plastic (65.7 tons/week), which is around 1.68 times more than the benchmark (39.0 tons/week). Among all optimization-based approaches, Myopic and Week perform the worst and comparably. This suggests that the benefit of being forward-looking of the Week implementation is outweighed by the path dependency issue (to which Week is more sensitive because it re-optimizes every week only). Accordingly, methods that re-optimize every day *and* consider a longer planning horizon, namely Week-Folding and Week-Rolling, have a clear edge.

Figure 7b breaks down these yearly average by season. We observe that the quantity of plastic collected (by any method) is higher in summer than in winter (53.3–87.0 range in summer vs. 21.9–



(a) Results averaged over the 13 simulations (b) Results averaged by seasons

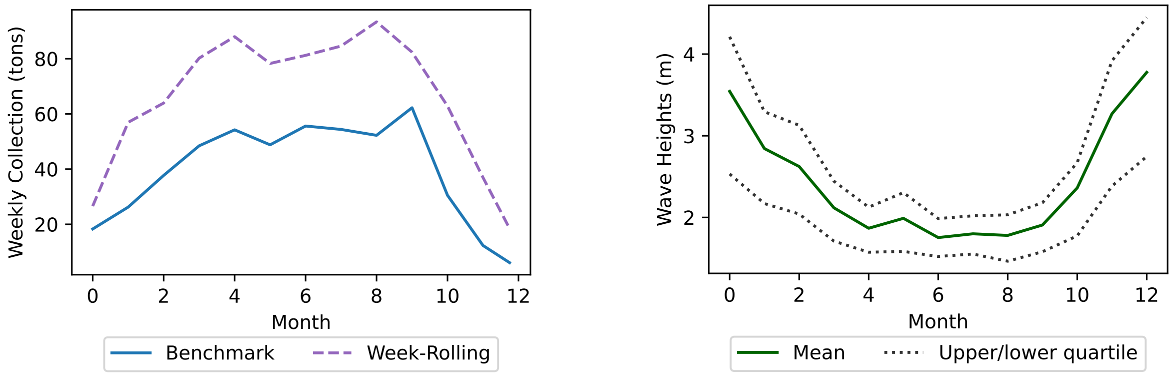
Figure 7 Weekly quantity of plastic collected, for the benchmark and each of the optimization-based approach. For the right panel, 'Winter' = January-March (3 first simulations) + October-December (3 last simulations); 'Summer' = April-September (7 simulations).

44.3 in winter) and we observe that the relative benefit from our optimization methods (e.g., Week-Rolling) compared with the benchmark is higher during these winter months ($\times 1.6$ in summer vs. $\times 2.0$ in winter). Similarly, the gain from using a 7-day (Week-Rolling) instead of 1-day (Myopic) planning horizon is negligible in summer (84.0 vs. 82.7: +1.6%) but most acute in winter (44.3 vs. 31.4: +41%). We investigate the mechanisms driving this pattern in the coming section.

5.3. Heterogeneity across seasons and impact of extraction scheduling

Figure 7b raises the question of the impact of season on the plastic collection efficiency. We should emphasize from the start that the behavior we observe is not driven by differences in the overall plastic density during the year. Values of plastic density continuously increase over the year (roughly by 10% in our 2002 data, by around 2% nowadays) but do not exhibit this inverted U-shape (see Figure E.4a in Section E.3).

If plastic density (i.e., the objective of our optimization problem) cannot explain this behavior, it is natural to consider the impact of weather (which drives most of the operational constraints) on the heterogeneous performance across seasons. We observe in Figure 8 that wave height (right panel) follows the same pattern as the quantity of plastic collected (left panel), with lower waves experienced in the middle of the year (April-August) and higher waves during November-February. High waves affect the collection process in two ways. First, the system cannot operate when the wave height exceeds 6 meters. Hence, the average 'collectable' plastic density is much lower in



(a) Plastic collected

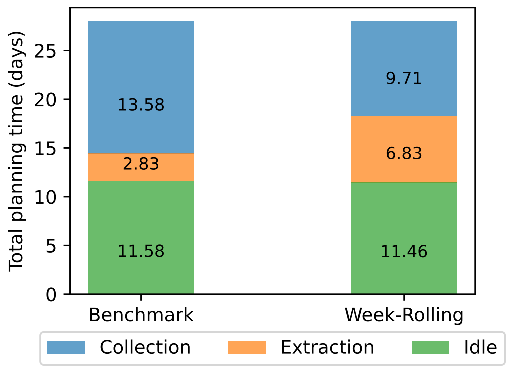
(b) Wave height

Figure 8 Plastic collected (left panel) and average wave height in the GPGP (right panel) across the year 2002.

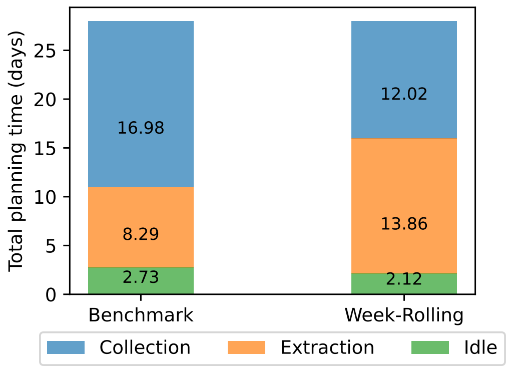
winter than in summer (see Figure E.4b in Section E.3). Furthermore, extractions require the waves to be below 2.5 meters for the first 6 hours, and below 3.5 meters for 12 hours. Hence, weather and its impact on the feasibility of extractions most likely drive the behavior we observe.

To confirm this intuition, we quantify the time spent by the system while waiting to extract. At any point in time, the system can be in one of three phases: it can be actively collecting plastic, it can be undergoing an extraction, or it can be idle (i.e., unable to collect plastic because it has reached capacity but unable to start an extraction either because of weather). For each 4-week simulation, we compute the number of days the system spent in collecting, extracting (which is equivalent to the number of extractions performed), and staying idle. For each simulation, the above three numbers should add up to 28 days. We averaged these numbers per season (winter/summer), where we define ‘winter’ as the first three and last three simulations and ‘summer’ as the remaining 7 ones. Figure 9 reports these metrics for the benchmark and Week-Rolling methods in both winter months (Figure 9a) and summer months (Figure 9b).

First, we observe that, in both winter and summer, Week-Rolling spends less time collecting than the benchmark. Given that Week-Rolling collects more plastic (67% more on average), this indicates that our approach is more efficient: it collects more in less time. Comparing Figure 9a and Figure 9b, we observe that idle time is significantly higher in winter, confirming the fact that weather conditions limit the ability to extract (hence, to collect further) during winter. Surprisingly, Week-Rolling does not materially reduce total idle time in winter (around 12 days out of 28 for both methods). However, Week-Rolling performs twice as many extractions, around 6.8 times on average compared with 2.8 times for the benchmark (which aligns with the increase in quantity



(a) Winter



(b) Summer

Figure 9 Number of days spent on collection, extraction and idle per 4-week simulation, for the benchmark and Week-Rolling methods. Results are aggregated over the winter (left panel) and summer (right panel) months.

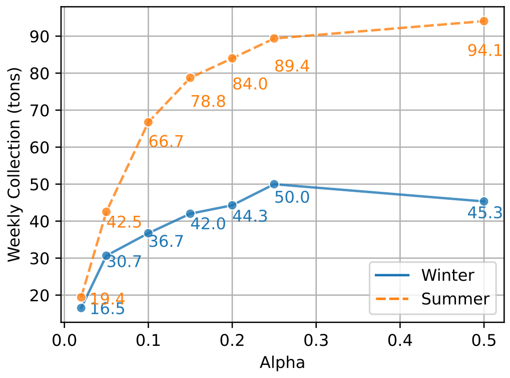
of plastic collected), so Week-Rolling experiences a much lower idle time per extraction than the benchmark.

The above observations highlight the importance of jointly finding a collection route and a schedule for the extractions for overall efficiency—and the importance of considering a sufficiently long planning horizon. On this matter, our 7-day optimization approach that can explicitly account for weather-related constraints experiences greater benefits in the winter, when the ability to extract constitutes the main bottleneck.

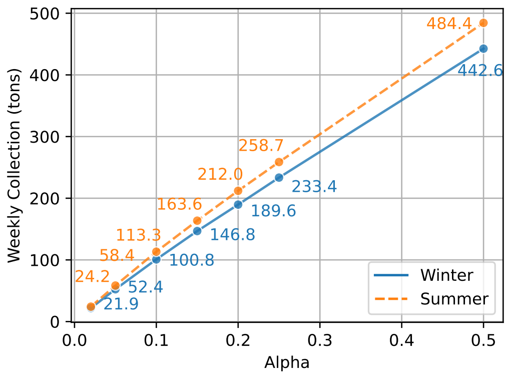
5.4. Designing a new system: is bigger better?

In Section 5.2, we show that with optimization, we can improve the collection speed from 40 tons/week to 60+ tons/week, using their current system. In this section, we use our optimization model to help answer strategic dimensioning decision for the next-generation system. The Ocean Cleanup was first considering increasing the span of the system from $0.6 \sim 0.8$ km ($\alpha \approx 0.1$) to $1.6 \sim 1.8$ km ($\alpha \approx 0.2$), *without increasing the size of the retention zone (25 tons)*. Indeed, the size of the retention zone is partially constrained in practice by the size of the ship used to store and sort the plastic collected.

Figure 10 represents the weekly quantity of plastic collected in winter (solid blue lines) and summer (dash orange lines), for increasing values of α, in the case of a 25-ton capacity (left panel, Figure 10a) and an infinite capacity (right panel, Figure 10b). Without capacity constraints from the retention zone, one expects the total quantity of plastic collected to depend linearly in the



(a) System capacity 25 tons



(b) System infinite capacity (no extraction)

Figure 10 Weekly quantity of plastic collected by DP (Rolling) under different α. Results are aggregated over the winter (blue solid line) and summer (orange dashed line) months, and with (left panel) or without (right panel) extraction.

span of the system α, as displayed in Figure 10b. However, with a finite capacity (Figure 10a), we observe (i) an overall lower quantity of plastic collected (which is due to the need to extract and the fact that we stop collection during extraction), and (ii) a strong concave dependency of the plastic collected on α. Indeed, by doubling the span size from $\alpha = 0.1$ to $\alpha = 0.2$, the weekly collection increases by 26% in summer (from 66.7 to 84.0 tons/week) and by 20% only in winter (from 36.7 to 44.3 tons/week). Moreover, increasing the span beyond $\alpha = 0.25$ (i.e., 2 km span) provides barely any improvement.

Intuitively, this different behavior across seasons is due to the fact that a larger span requires more frequent extractions, which are very sensitive to weather conditions. The comparison between Figure 10a and Figure 10b highlights the impact of having a finite-capacity retention zone on the overall performance. In the future, the difficulty to extract could largely erode the benefit of having a larger system. This leads us to the next question: how to design a new system with better extraction?

There are several ways to improve the current extraction process. One solution could be to reduce the unit extraction time, namely, the time spent per extraction. In practice, this could be achieved via more efficient extraction operations. For example, one could first empty the plastic from the retention zone on the deck (taking approximatively 6 hours), put the system back into the water, and sort the plastic while resuming the plastic collection. Another solution could be to increase the total capacity of the retention zone, which we do not discuss in this paper.

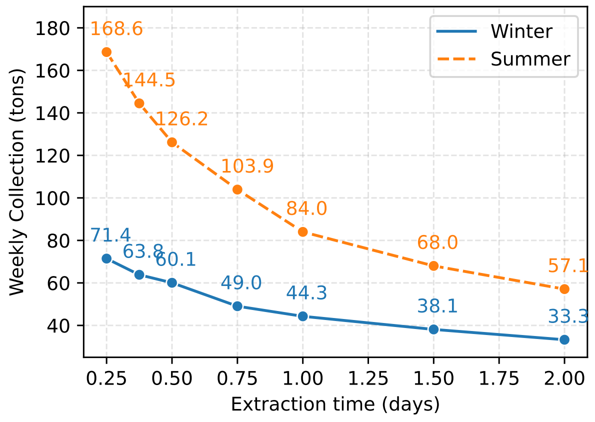


Figure 11 Weekly collections of DP (Rolling) with different unit extraction times. Results are aggregated over the winter (blue solid line) and summer (orange dashed line) months.

Figure 11 represents the weekly quantity of plastic collected in winter (solid blue lines) and summer (dash orange lines), for increasing values of unit extraction time, in the case of a 25-ton capacity. By reducing the time per extraction from 1 day (current practice) to 0.25 days (or 6 hours), the weekly collection increases by 101% in summer (84.0 to 168.6 tons/week) and by 61% in winter (44.3 to 71.4 tons/week). Observe, in comparison, that the potential improvement by further increasing the span size beyond $\alpha = 0.2$ (Figure 10a) is less than 12%. Reducing the unit extraction time demonstrates a greater potential for impact, both in winter and summer.

We emphasize that the above improvement solely comes from a shorter extraction time, not from a lower impact of weather constraints since we kept, in our implementation, the same weather constraints for extraction (described in the EC, Section A.1) irrespective of the extraction time. In practice, shorter extraction time might also translate into less stringent weather constraints, which could in turn provide additional benefits.

6. Conclusion

Our oceans are being threatened by the millions of tons of plastic that have been emitted over the recent decades. To limit future harm to marine ecosystems and activities, we need to clean up oceans from plastic, as quickly as possible. To this end, we develop a graph-based model and formulate the problem of routing a plastic-collecting system in the GPGP to maximize the quantity of plastic encountered as a longest path problem. However, due to the plastic dynamics and the direct impact of collection on these dynamics, our resulting longest path problem (4) is non-convex and non-separable over edges. To deal with these computational difficulties, we propose to relax the reward dynamics and solve large-scale instances of this relaxation in linear time using a DP algorithm. Then, we obtain near-optimal solutions to our original problem, together with certificates of near optimality, by building an efficient search algorithm based on geographical clustering and the terminal values of the DP algorithm (and not only on its optimal solution). We also develop a tailored branch-and-bound algorithm which solves instances with 24 timesteps (3 days) to optimality in minutes and instances with 56 timesteps (7 days) within 4–5% in an hour.

On one-year weather and plastic density data, we observe that our optimization approaches increase the quantity of plastic collected by 68% compared with the status quo, thus accelerating the path to plastic-free oceans. We also leverage our optimization algorithms to explore the non-linear relationships between system characteristics and system performance. For example, because of difficulties to extract (i.e., empty the capacity of the system) in winter, we find that increasing the span of the system beyond 1.8 km will have barely any impact on collection efficiency in winter.

For our current application, the main concern and area for future research is to account for uncertainty in weather predictions and plastic dynamics. In particular, we are currently investigating whether collection of real-world data by drones or satellites could help quantify uncertainty and lead to robust versions of our longest path problem. More broadly, we are excited to study whether the class of longest path problems we identify in (4) could find other applications, as a model for operations with nature dynamics.

Endnotes

1. The dynamics in (3) implicitly assume that we can decompose the time interval $[t, t+1)$ into two distinct steps: a first step where some of the plastic present at time t is removed, and a second step where the remaining plastic float according to the dynamics captured by $\boldsymbol{Q}^t$. Of course, this is a simplification of reality where these two steps occur concurrently. Yet, we believe it is an appropriate model of reality, which captures the essence of path dependency.

2. Since each 4-week simulation requires 5 weeks of data (because some of our optimization algorithms are forward-looking and take into account information of the next 7 days), we would need 53 weeks to conduct 13 non-overlapping 28-day simulations. Instead, we start the last experiment one week earlier (on day 329 instead of 336).

Appendix. Dynamic Programming Algorithm for the Longest Path Problem

In this section, we provide the pseudo-code of an efficient DP algorithm for solving longest-path problems, given that our graph $\mathcal{G}$ is a DAG. The algorithm is described in Algorithm 1.

At each iteration, the algorithm finds the longest path ending in $s \in \mathcal{S}_t$ by connecting the longest path between $\mathcal{S}_0$ and s' and the edge (s', s) for some $s' \in \mathcal{S}_t$. The corresponding maximization problem can be solved by exhaustively searching through $\mathcal{S}_t$. Fortunately, in our graph $\mathcal{G}$, because of the no-sharp-angle constraint, we have $|\{s' \in \mathcal{S}_t : s \in \text{succ}(s')\}| \leq 3$, so this maximization problem can be solved in $\mathcal{O}(1)$ operations.

Algorithm 1: Dynamic Programming Algorithm for Finding the Longest Path

```
   Data: Weighted graph $\mathcal{G}$ with weight $\{w_{s,s'}\}_{(s,s')\in\mathcal{E}}$;
1  Initialize (values and optimal paths): $V^0(s) = 0$, path$[s] = \{s\}$, for all $s \in \mathcal{S}_0$;
2  for $t$ = 1:T do
3      for $s \in \mathcal{S}_t$ do
4          Find an optimal previous state, $s^* \in \arg\max_{s' \in \mathcal{S}_{t-1}: s \in \text{succ}(s')} \{w_{s',s} + V^{t-1}(s')\}$;
5          Update value function: $V^t(s) = w^t_{s^*,s} + V^{t-1}(s^*)$ ;
6          Update optimal path: path$[s] \leftarrow$ path(path$[s^*], s)$;
7      end
8  end
9  Find an optimal terminal state $s^\star \in \arg\max_{s \in \mathcal{S}_T} V^T(s)$;
10 Return: value $V^T(s^\star)$, longest path path$[s^\star]$.
```

References

Appelgren LH (1971) Integer programming methods for a vessel scheduling problem. *Transportation Science* 5(1):64–78.

Aydin N, Lee H, Mansouri SA (2017) Speed optimization and bunkering in liner shipping in the presence of uncertain service times and time windows at ports. *European Journal of Operational Research* 259(1):143–154.

Battarra M, Pessoa AA, Subramanian A, Uchoa E (2014) Exact algorithms for the traveling salesman problem with draft limits. *European Journal of Operational Research* 235(1):115–128.

Bellman R (1958) On a routing problem. *Quarterly of Applied Mathematics* 16(1):87–90.

Björklund A, Husfeldt T, Khanna S (2004) Approximating longest directed paths and cycles. *International Colloquium on Automata, Languages and Programming* 3142:222–233.

Bulterman RW, van der Sommen FW, Zwaan G, Verhoeff T, van Gasteren AJM, Feijen WHJ (2002) On computing a longest path in a tree. *Information Processing Letters* 81(2):93–96.

Chen C, Chen XQ, Ma F, Xiao-Jun Zeng, Zeng XJ, Wang J (2019) A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Engineering* 189:106299.

Cheng Y, Zhang W (2018) Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* 272:63–73.

Christiansen M, Fagerholt K, Ronen D (2004) Ship routing and scheduling: Status and perspectives. *Transportation Science* 38(1):1–18.

Church R, ReVelle C (1974) The maximal covering location problem. *Papers of the Regional Science Association* 32(1):101–118.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2022) *Introduction to Algorithms* (MIT press).

Danna E, Rothberg E, Pape CL (2005) Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming* 102:71–90.

Dantzig GB (1960) On the shortest route through a network. *Management Science* 6(2):187–190.

de Vries R, Egger M, Mani T, Lebreton L (2021) Quantifying floating plastic debris at sea using vessel-based optical data and artificial intelligence. *Remote Sensing* 13(17):3401.

de Wit C (1990) Proposal for low cost ocean weather routeing. *The Journal of Navigation* 43(3):428–439.

Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

Dijkstra H, van Beukering P, Brouwer R (2021) In the business of dirty oceans: Overview of startups and entrepreneurs managing marine plastic. *Marine Pollution Bulletin* 162:111880.

Fagerholt K, Christiansen M (2000) A travelling salesman problem with allocation, time window and precedence constraints — an application to ship scheduling. *International Transactions in Operational Research* 7(3):231–244.

Ford LRJ (1956) Network flow theory. Technical report, Rand Corp Santa Monica Ca.

Gall SC, Thompson RC (2015) The impact of debris on marine life. *Marine Pollution Bulletin* 92(1-2):170–179.

Geyer R, Jambeck JR, Law KL (2017) Production, use, and fate of all plastics ever made. *Science Advances* 3(7):e1700782.

Granado I, Hernando L, Galparsoro I, Gabiña G, Groba C, Prellezo R, Fernandes JA (2021) Towards a framework for fishing route optimization decision support systems: Review of the state-of-the-art and challenges. *Journal of Cleaner Production* 320:128661.

Groba C, Sartal A, Bergantiño G (2020) Optimization of tuna fishing logistic routes through information sharing policies: A game theory-based approach. *Marine Policy* 113:103795.

Groba C, Sartal A, Vázquez XH (2015) Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *Computers & Operations Research* 56:22–32.

Groba C, Sartal A, Vázquez XH (2018) Integrating forecasting in metaheuristic methods to solve dynamic routing problems: Evidence from the logistic processes of tuna vessels. *Engineering Applications of Artificial Intelligence* 76:55–66.

Hagiwara H, Spaans JA (1987) Practical weather routing of sail-assisted motor vessels. *The Journal of Navigation* 40(1):96–119.

Ioannidou K, Nikolopoulos SD (2013) The longest path problem is polynomial on cocomparability graphs. *Algorithmica* 65(1):177–205.

James RW (1957) *Application of Wave Forecasts to Marine Navigation* (New York University).

Kaandorp ML, Lobelle D, Kehl C, Dijkstra HA, van Sebille E (2023) Global mass of buoyant marine plastics dominated by large long-lived debris. *Nature Geoscience* 16(8):689–694.

Karger D, Motwani R, Ramkumar GDS (1997) On approximating the longest path in a graph. *Algorithmica* 18(1):82–98.

Karp RM (2010) *Reducibility Among Combinatorial Problems* (Springer).

Klink D, Peytavin A, Lebreton L (2022) Size dependent transport of floating plastics modeled in the global ocean. *Frontiers in Marine Science* 9:903134.

Kosmas OT, Vlachos D (2012) Simulated annealing for optimal ship routing. *Computers & Operations Research* 39(3):576–581.

Langbein J, Stelzer R, Frühwirth T (2011) A rule-based approach to long-term routing for autonomous sailboats. *Robotic Sailing: Proceedings of the 4th International Robotic Sailing Conference*, 195–204 (Springer).

Lebreton L, Slat B, Ferrari FF, Sainte-Rose B, Aitken J, Marthouse R, S Hajbane, Hajbane S, Cunsolo S, Schwarz A, Levivier A, Noble K, Debeljak P, Maral H, Schoeneich-Argent R, Brambini R, Reisser J (2018) Evidence that the Great Pacific Garbage Patch is rapidly accumulating plastic. *Scientific Reports* 8(1):4666–4666.

Lebreton L, Van Der Zwet J, Damsteeg JW, Slat B, Andrady AL, Reisser J (2017) River plastic emissions to the world's oceans. *Nature Communications* 8(1):15611–15611.

Li WC, Tse H, Fok L (2016) Plastic waste in the marine environment: A review of sources, occurrence and effects. *Science of the Total Environment* 566:333–349.

Madraki G, Judd RP (2019) Recalculating the length of the longest path in perturbed directed acyclic graph. *IFAC-PapersOnLine* 52(13):1560–1565.

Malaguti E, Martello S, Santini A (2018) The traveling salesman problem with pickups, deliveries, and draft limits. *Omega* 74:50–58.

Meijer LJ, Van Emmerik T, Van Der Ent R, Schmidt C, Lebreton L (2021) More than 1000 rivers account for 80% of global riverine plastic emissions into the ocean. *Science Advances* 7(18):eaaz5803.

Meng Q, Wang T (2011) A scenario-based dynamic programming model for multi-period liner ship fleet planning. *Transportation Research Part E: Logistics and Transportation Review* 47(4):401–413.

Moore EF (1959) The shortest path through a maze. *Proceedings of International Symposium on the Theory of Switching, 1959*, 285–292.

Pandit SN (1962) Some observations on the routing problem. *Operations Research* 10(5):726–727.

Park YJ, Garaba S, Sainte-Rose B, Han HJ (2022) Satellite remote sensing of marine litter floating in open ocean and coastal waters. *Living Planet Symposium 2022* (The European Space Agency (ESA)).

Park YJ, Garaba SP, Sainte-Rose B (2021) Detecting the great pacific garbage patch floating plastic litter using worldview-3 satellite imagery. *Optics Express* 29(22):35288–35298.

Peytavin A, Sainte-Rose B, Forget G, Campin JM (2021) Ocean plastic assimilator v0. 2: assimilation of plastic concentration data into lagrangian dispersion models. *Geoscientific Model Development* 14(7):4769–4780.

Pollack M, Wiebenson W (1960) Solutions of the shortest-route problem-a review. *Operations Research* 8(2):224–230.

Robinson NM, Nelson WA, Costello MJ, Sutherland JE, Lundquist CJ (2017) A systematic review of marine-based species distribution models (SDMs) with recommendations for best practice. *Frontiers in Marine Science* 4:421.

Sainte-Rose B, Rakotonirina AD, van den Bremer T, Pham Y (2022) Numerical simulation of the wave-induced drift of floating marine plastic debris modeled as discrete particles. *International Conference on Offshore Mechanics and Arctic Engineering*, volume 85925, V007T08A008 (American Society of Mechanical Engineers).

Schmidt C, Krauth T, Wagner S (2017) Export of plastic debris by rivers into the sea. *Environmental Science & Technology* 51(21):12246–12253.

Schnurr RE, Alboiu V, Chaudhary M, Corbett RA, Quanz ME, Sankar K, Srain HS, Thavarajah V, Xanthos D, Walker TR (2018) Reducing marine pollution from single-use plastics (sups): A review. *Marine Pollution Bulletin* 137:157–171.

Schrijver A (2012) On the history of the shortest path problem. *Documenta Mathematica* 17(1):155–167.

Sen D, Padhy CP (2015) An approach for development of a ship routing algorithm for application in the North Indian Ocean region. *Applied Ocean Research* 50:173–191.

Shao W, Zhou P, Thong SK (2012) Development of a novel forward dynamic programming method for weather routing. *Journal of Marine Science and Technology* 17(2):239–251.

Shimbel A (1954) Structure in communication nets. *Proceedings of the Symposium on Information Networks*, 119–203 (Polytechnic Institute of Brooklyn).

Skoglund L (2012) A new method for robust route optimization in ensemble weather forecasts. Master thesis, KTH, School of Engineering Sciences (SCI).

Sniedovich M (2006) Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics* 35(3):599–620.

Stalhane M, Hvattum LM, Skaar V (2015) Optimization of routing and scheduling of vessels to perform maintenance at offshore wind farms. *Energy Procedia* 80:92–99.

Stokes GG (1847) On the theory of oscillatory waves. *Transactions of the Cambridge Philosophical Society* 8:441–455.

Takashima K, Mezaoui B, Shoji R (2009) On the fuel saving operation for coastal merchant ships using weather routing. *TransNav, International Journal on Marine Navigation and Safety of Sea Transportation* 3(4):401–406.

Taylor GI (1915) I. Eddy motion in the atmosphere. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 215(523-537):1–26.

Ting SC, Tzeng GH (2003) Ship scheduling and cost analysis for route planning in liner shipping. *Maritime Economics & Logistics* 5(4):378–392.

Uehara R, Uno Y (2007) On computing longest paths in small graph classes. *International Journal of Foundations of Computer Science* 18(05):911–930.

Vettor R, Tadros M, Ventura M, Guedes Soares C (2016) Route planning of a fishing vessel in coastal waters with fuel consumption restraint. *Maritime Technology and Engineering* 3:167–173.

Wilcox C, Van Sebille E, Hardesty BD (2015) Threat of plastic pollution to seabirds is global, pervasive, and increasing. *Proceedings of the National Academy of Sciences* 112(38):11899–11904.

Winterstetter A, Grodent M, Kini V, Ragaert K, Vrancken KC (2021) A review of technological solutions to prevent or reduce marine plastic litter in developing countries. *Sustainability* 13(9):4894.

Yoon H, Nguyen V, Nguyen T (2018) Development of solution for safe ship considering seakeeping performance. *TransNav, International Journal on Marine Navigation and Safety of Sea Transportation* 12(3):517–525.

Zheng J, Zhang H, Yin L, Liang Y, Wang B, Li Z, Song X, Zhang Y (2019) A voyage with minimal fuel consumption for cruise ships. *Journal of Cleaner Production* 215:144–153.

Zis T, Psaraftis HN, Ding L (2020) Ship weather routing: A taxonomy and survey. *Ocean Engineering* 213:107697.

Zoppoli R (1972) Minimum-time routing as an N-stage decision process. *Journal of Applied Meteorology (1962-1982)* 11(3):429–435.

Electronic Companion: Optimizing the Path Towards Plastic-Free Oceans

This appendix will be published as a separate Electronic Companion.

A Graph-Based Optimization Formulation

A.1. List of all operational and weather constraints

In Table A.1, we summarize all the operational and weather constraints The Ocean Cleanup encountered in their practice.

Constraints	Description
Speed	The propelling speed cannot exceed 1.5 knots.
Angle	The system can turn at most $45°$ within 3 hours.
Wave (direction)	If wave height > 4.5m, head against the direction of the wave.
Wave (screen)	If wave height > 6m, the screen cannot work, no plastic is collected.
Wave (extraction 1)	For the first 6 hours of each extraction, wave height ≤ 2.5m.
Wave (extraction 2)	For the remaining 18 hours of extraction, at least 12 hours with height ≤ 3.5m.
Wind (extraction)	If wind speeds > 25 knots, the crane cannot work.

Table A.1 Operational and weather constraints for The Ocean Cleanup's system 002/B

Note that most of these constraints depend on the location and the steering direction at time t. This is the reason why we introduced redundancy in the information used to describe the trajectory of the system at time t in Section 3.1, i.e., to efficiently account for the different constraints in our problem. Essentially, the sequence of locations $\{\ell_t\}_t$ is sufficient to describe the system's trajectory and providing both ℓ_t and d_t is equivalent to providing ℓ_t and ℓ_{t-1}.

A.2. Directed acyclic graph corresponding to Figure 3

Figure A.1 shows the graph corresponding to the example of Figure 3. Here, $\mathcal{S}_0 = \{((1,1), \searrow, 0)\}$. The dashed orange nodes (edges) represent the states (transitions) forbidden by weather-related constraints.

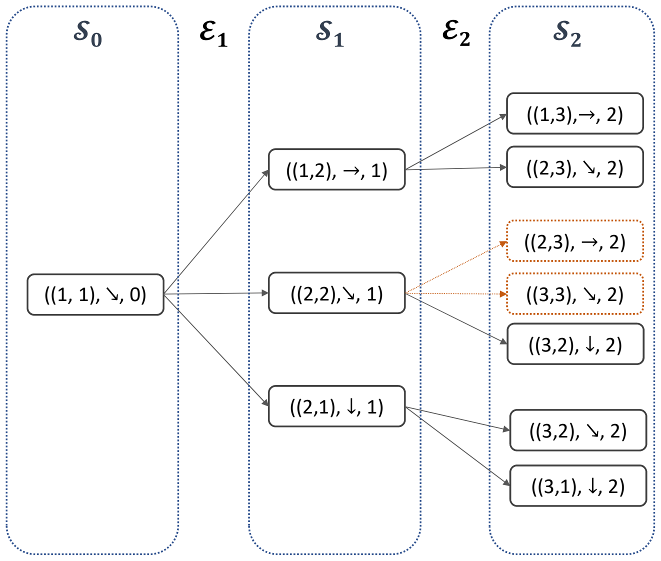


Figure A.1 The structure of graph in the example of Figure 3. Edges and nodes in orange dashes are forbidden by the weather constraints.

A.3. Extended graph representation for modeling extraction scheduling decisions

In Section 3.1, we present a graph-based modeling to describe the trajectory of the system as a path in an appropriate graph. In this section, we expand our description of the state of the system to incorporate the decision of extraction scheduling, while maintaining our efficient longest path interpretation. As described in Section 1.1, the plastic collected accumulates in a retention zone with fixed capacity and that needs to be emptied regularly. In addition, this emptying process, called extraction, requires the weather condition to be favorable (in particular, wave height need to be below 2.5 meters). Accordingly, we need to enrich the state space to include the type of activity the system is performing (collection or extraction) as well as an indication of the load of the retention zone.

Remark EC.1. A depot is something we could incorporate as well. Actually, performing an extraction can be seen as the system going to an abstract type of 'depot' (an action that has to be done regularly and has to be done in specific locations). Therefore, modeling techniques similar to the ones presented in this section could be used to account for the need to regularly visit a depot location.

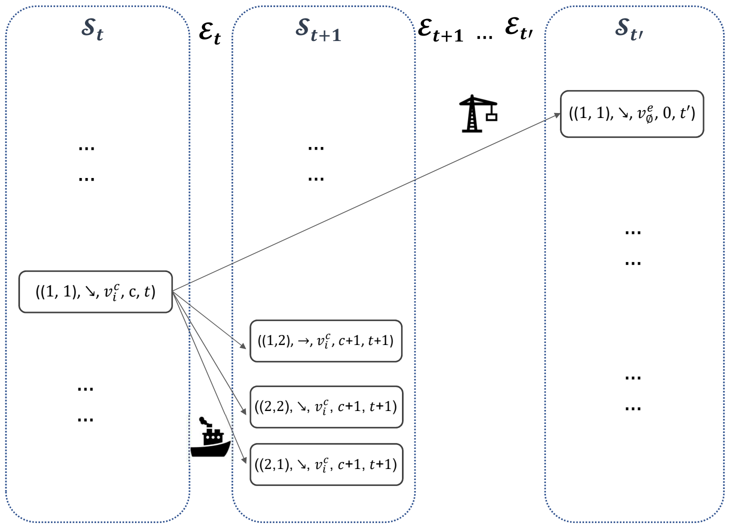
The system can perform two types of activity: it can either be collecting plastic or extracting plastic from the retention zone. Actually, because the system cannot extract when weather conditions are unfavorable, the system can also be idle, i.e., not actively collecting because the retention zone is full but not extracting either. Accordingly, we define five possible activities, denoted $\mathcal{V} = \{v_1^c, v_\emptyset^i, v_1^i, v_\emptyset^e\}$, where the subscript indicates the propelling speed of the system ($\emptyset$ for not moving and 1 for moving at normal speed) and the superscript indicates the activity (*c*ollection, *i*dle, and *e*xtraction). Here, we assume that the system does not move during an extraction (because of operations). However, when idle, it can either stay at the same location and wait for the weather to improve or move towards an area with nicer weather. We denote ΔT_e the time required for extraction (in time periods). By construction, we will ensure that the status $v_\emptyset^e$ corresponds to the extraction being finished, since complexity of the extraction process does not provide the flexibility to perform partial extractions. In addition, the time ΔT_e includes extraction and time needed to re-direct the system so the no-sharp-turn constraint does not apply between the steering directions before and after extraction.

In addition to v, we introduce a state dimension to capture the current load of the retention zone. More precisely, we divide the capacity of the retention zone into C increments, corresponding to the typical quantity of plastic collected during one time period. Hence, we can introduce a variable $c \in \mathcal{C} := \{0, 1, \ldots, C\}$ capturing load of the retention zone.

With these additional variables, we can describe the state of the system as a 5-tuple $s = (\ell, d, v, c, t) \in \mathcal{L} \times \mathcal{D} \times \mathcal{V} \times \mathcal{C} \times \mathcal{T}$ and we can represent the set of all possible joint trajectories and extraction schedules as a path in a graph, whose nodes and edges are constructed recursively thanks to a successor operator. For this state s, an admissible next state $s' = (\ell', d', v', c', t')$ should satisfy the following constraints:

- If the retention zone is empty and the system is currently extracting (and is done emptying), then the system resumes collection and can go in any direction. Formally, if $c = 0$ and $v = v_\emptyset^e$, we have $s' = (\ell', d', v_1^c, 1, t+1)$ where (ℓ', d') satisfies all constraints defining $\mathrm{succ}(\ell, d, t)$ except for the no-sharp-turn constraint.

- If the retention zone is not full and the system is currently collecting, then the system can either continue collecting (and obey the same constraints as those described in Section 3.1) or proactively start an extraction (if feasible). Formally, if $c < C$ and $v = v_1^c$, we have either $s' = (\ell', d', v_1^c, c+1, t+1)$ with $(\ell', d', t+1) \in \mathrm{succ}(\ell, d, t)$ or $s' = (\ell, d, v_0^e, 0, t + \Delta T_e)$ if possible to empty at location ℓ.



$\mathcal{S}_t$
$\mathcal{E}_t$
$\mathcal{S}_{t+1}$
$\mathcal{E}_{t+1}$ … $\mathcal{E}_{t'}$
$\mathcal{S}_{t'}$
$((1,1), \searrow, v^e_\emptyset, 0, t')$
$((1,1), \searrow, v^c_i, c, t)$
$((1,2), \rightarrow, v^c_i, c+1, t+1)$
$((2,2), \searrow, v^c_i, c+1, t+1)$
$((2,1), \searrow, v^c_i, c+1, t+1)$

Figure A.2 Successor of a state in a graph that includes both collection and extraction

- If the retention zone is currently full, then the next state can either be idle or extraction. Formally, if $c = C$, then s' must fall in one of the three cases: $(\ell, d, v^i_\emptyset, C, t+1)$, $(\ell', d', v^i_1, C, t+1)$ with $(\ell', d', t+1) \in \text{succ}(\ell, d, t)$, or $(\ell, d, v^e_\emptyset, 0, t + \Delta T_e)$ if extraction is possible at location ℓ.

As displayed in Figure A.2, compared with the simple model presented in Section 3.1, this extended state space and successor operator allows the system to "jump" some time periods: when deciding to extract, we directly connect the state of the system at the beginning of the extraction with the state of the system after extraction, i.e., ΔT_e time periods later. Nonetheless, the resulting graph $\mathcal{G}$ conserves the good properties we leverage in our algorithm: First, $\mathcal{G}$ is a DAG, because all edges are of the form $(s, s') \in \mathcal{S}_t \times \mathcal{S}_{t'}$ with $t' > t$, hence ensuring the correctness of the DP algorithm for longest path problems. Second, for a given state s, its number of successors (or predecessors) is uniformly bounded by a constant.

A.4. Longest path and network flow optimization problems

Mathematically, the longest path problem (1) can be formulated as a mixed-integer optimization problem by introducing binary variables $x^t_s \in \{0,1\}$ indicating whether the system is in state s at time t. The resulting optimization writes as follows:

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}_t} r^t_s x^t_s,$$

with

$$\mathcal{X} := \left\{ \boldsymbol{x} \in \{0,1\}^{\mathcal{S}} \,\middle|\, \begin{array}{ll} \sum_{s\in\mathcal{S}_t} x_s^t = 1, & \forall\, t \in \mathcal{T}, \\ \sum_{s'\in \mathrm{succ}(s)} x_{s'}^{t+1} \geq x_s^t, & \forall\, t\in\mathcal{T}, s \in \mathcal{S}_t \end{array} \right\}.$$

Despite its compactness, we should emphasize that this optimization formulation is not the most computationally efficient because of the integrality constraints. Alternatively, the optimization problem above can be seen as a special case of a network flow optimization problem for which ideal formulations are well known.

Formally, let us now view any sequence of states $\{s_t\}_{t\in\mathcal{T}}$ as binary flow variables $\boldsymbol{f} \in \{0,1\}^{\mathcal{E}}$ where $f_e = 1$ if and only if $e = (s_t, s_{t+1})$ for some $t \in \mathcal{T}$. In other words, $\boldsymbol{f}$ indicates the transitions that the system goes through, while $\boldsymbol{x}$ is encoding the system states. Naturally, one can recover the flow variables from the state variables, and vice versa, via the relations

$$\begin{aligned} f_{s,s'} &= x_s^t x_{s'}^{t+1}, && \forall (s,s') \in \mathcal{E}_t, \\ x_s^t &= \sum_{s'\in\mathrm{succ}(s)} f_{s,s'}, && \forall s \in \mathcal{S}_t. \end{aligned}$$

With these notations, the DP algorithm, Algorithm 1, can solve any optimization problem of the form

$$\max_{\boldsymbol{f}\in\mathcal{F}} \sum_{e\in\mathcal{E}} w_e f_e, \tag{EC.1}$$

where $\mathcal{F}$ is the set of admissible binary flows in $\mathcal{G}$:

$$\mathcal{F} := \left\{ \boldsymbol{f} \in \{0,1\}^{\mathcal{E}} \,\middle|\, \begin{array}{l} \sum_{(s,s')\in\mathcal{E}_0} f_{s,s'} = 1, \\ \sum_{s' : s \in \mathrm{succ}(s')} f_{s',s} = \sum_{s'\in\mathrm{succ}(s)} f_{s,s'} \;\; \forall\, t\in\mathcal{T}, s\in\mathcal{S}_t \end{array} \right\}.$$

The first constraint is the initial condition and ensures that the path starts at some state $s \in \mathcal{S}_0$. The second set of constraints corresponds to the common flow conservation constraints, ensuring that, at each node s, inflow equals outflow. Together, they imply that $f_e \leq 1$ for any $e \in \mathcal{E}$. So, the binary constraints can be replaced by an integrality constraint, $\boldsymbol{f} \in \mathbb{Z}_+^{\mathcal{E}}$. Furthermore, it is well known that the polytope of unconstrained flows in a graph is integral, i.e., its extreme points are integral flows. Hence, without loss of optimality, we can assume that the optimization problem (1) occurs over the convex set

$$\mathcal{F} = \left\{ \boldsymbol{f} \in \mathbb{R}_+^{\mathcal{E}} \,\middle|\, \begin{array}{l} \sum_{(s,s')\in\mathcal{E}_0} f_{s,s'} = 1, \\ \sum_{s' : s \in \mathrm{succ}(s')} f_{s',s} = \sum_{s'\in\mathrm{succ}(s)} f_{s,s'} \;\; \forall\, t\in\mathcal{T}, s\in\mathcal{S}_t \end{array} \right\}.$$

B Analytical Analysis of the Path-Dependent Reward

In this section, we supplement the analysis of path-dependent rewards from Section 4.1, by eliciting the dynamics operator $\mathcal{Q}^t$ in (4), the proof of Proposition 1, and deriving the second-order expansion (in α) of the path-dependent reward $\boldsymbol{r}_{|\boldsymbol{x}}$.

B.1. Path dependency for state-based rewards

In our optimization formulations, we use binary variables that are defined over states, $\tilde{\boldsymbol{x}} \in \{0,1\}^{\cup_t \mathcal{S}_t}$, while plastic and reward dynamics are naturally expressed using location indicators $\boldsymbol{x}^t \in \{0,1\}^{\mathcal{L}}$—for clarity, in this section, we will systematically use $\tilde{\cdot}$ to indicate variables defined at a state level. The latter can be recovered from the former via the linear relationship

$$x_\ell^t = \sum_{s'=(\ell',d',t)\in\mathcal{S}_t:\ell'=\ell} \tilde{x}_{s'}^t,$$

which we write in matrix form $\boldsymbol{x}^t = \boldsymbol{W}^t \tilde{\boldsymbol{x}}^t$ for some matrix $\boldsymbol{W}^t \in \{0,1\}^{\mathcal{L}\times\mathcal{S}_t}$. With these notations, the reward dynamics (3) can be expressed as a function of $\tilde{\boldsymbol{x}}$ as

$$\boldsymbol{r}_{|\boldsymbol{x}^{0:t}}^{t+1} = \boldsymbol{Q}^t \left[\boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}}^t - \alpha\, \boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}}^t \circ (\boldsymbol{W}^t \tilde{\boldsymbol{x}}^t)\right].$$

We now need to map rewards defined for each location into rewards defined for each state. For example, if we assign to each state the reward associated with its location, we have $\tilde{\boldsymbol{r}} = (\boldsymbol{W}^t)^\top \boldsymbol{r}$. In practice, we consider a slightly different mapping where the reward for a state equals the reward of its location if the state is collecting, and 0 if the state is idle or extraction. In any case, we have a linear mapping of the form $\tilde{\boldsymbol{r}} = \boldsymbol{V}^t \boldsymbol{r}$ with $\boldsymbol{V}^t \in \{0,1\}^{\mathcal{S}_t\times\mathcal{L}}$ and $\boldsymbol{V}^t \le (\boldsymbol{W}^{t+1})^\top$. Similarly, we can obtain the reverse mapping $\boldsymbol{r} = \boldsymbol{U}^t \tilde{\boldsymbol{r}}$ with $\boldsymbol{U} \in \{0,1\}^{\mathcal{L}\times\mathcal{S}_t}$, i.e., for each location ℓ, $\boldsymbol{U}^t$ maps ℓ with one state $s \in \mathcal{S}_t$ whose reward is the reward at location ℓ (e.g., a collecting state currently at location ℓ).

Consequently, we are interested in maximizing

$$\sum_{t\in\mathcal{T}} \sum_{s\in\mathcal{S}_t} \tilde{r}_{s|\tilde{x}^{0:t-1}}^t \tilde{x}_s^t,$$

where the path-dependent rewards (defined at a state-level) $\tilde{\boldsymbol{r}}_{|\tilde{\boldsymbol{x}}^{0:t}}^{t+1}$ evolve according to the following dynamics:

$$\tilde{\boldsymbol{r}}_{|\tilde{\boldsymbol{x}}^{0:t}}^{t+1} = \boldsymbol{V}^{t+1} \underbrace{\boldsymbol{Q}^t \left[\boldsymbol{U}^t \tilde{\boldsymbol{r}}_{|\tilde{\boldsymbol{x}}^{0:t-1}}^t - \alpha \left(\boldsymbol{U}^t \tilde{\boldsymbol{r}}_{|\tilde{\boldsymbol{x}}^{0:t-1}}^t\right) \circ (\boldsymbol{W}^t \tilde{\boldsymbol{x}}^t)\right]}_{\boldsymbol{r}_{|\boldsymbol{x}^{0:t}}^{t+1}} =: \mathcal{Q}^t(\tilde{\boldsymbol{r}}_{|\tilde{\boldsymbol{x}}^{0:t-1}}, \tilde{\boldsymbol{x}}^t).$$

Finally, observe that the matrices $\boldsymbol{V}^t$ have non-negative entries. Accordingly, it preserves vector ordering, i.e., $\boldsymbol{r}^1 \leq \boldsymbol{r}^2 \implies \boldsymbol{V}^t\boldsymbol{r}^1 \leq \boldsymbol{V}^t\boldsymbol{r}^2$. Hence, upper and lower bounds on path-dependent rewards in Proposition 1 directly translate to rewards defined for each state.

B.2. Order-two expansion of the path-dependent reward

Lemma 1 provides the complete expansion (in α) of the path-dependent reward $\boldsymbol{r}_{|\boldsymbol{x}^{0:t}}$ when the plastic is static. In the general case, we need to account for plastic movements over time as described by the matrices $\boldsymbol{Q}^t$'s. We now derive analytically the first- and second-order term of this expansion in the general case, which is sufficient to provide intuition on the full expansion. For any $s \geq t$, we denote $\boldsymbol{Q}^{s:t} := \boldsymbol{Q}^s \times \boldsymbol{Q}^{s-1} \times \cdots \times \boldsymbol{Q}^t$. Hence, $\boldsymbol{Q}^t$ captures the plastic dynamics between time t and $t+1$, while $\boldsymbol{Q}^{s:t}$ captures the plastic dynamics from t to $s+1$. With these notations, we have:

Proposition EC.1. *The second-order expansion of the path-dependent reward $\boldsymbol{x}$ is given by:*

$$\begin{aligned}\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} =& \boldsymbol{r}^{t+1} \\ & - \alpha \sum_{0\leq t_1 \leq t} \boldsymbol{Q}^{t:t_1}\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right) \\ & + \alpha^2 \sum_{0 \leq t_2 < t_1 \leq t} \boldsymbol{Q}^{t:t_1}\left\{\left[\boldsymbol{Q}^{t_1-1:t_2}\left(\boldsymbol{r}^{t_2} \circ \boldsymbol{x}^{t_2}\right)\right] \circ \boldsymbol{x}^{t_1}\right\} \\ & + \mathcal{O}(\alpha^3).\end{aligned}$$

Proof of Proposition EC.1 When $t=0$, we already shown that the first-order expansion is exact and the second-order correction is 0, so the result holds.

Let us assume that the above expansion holds for $\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$, for some $t \geq 0$, then we have:

$$\boldsymbol{r}^{t+2}_{|x^{0:t+1}} = \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|x^{0:t}} - \alpha \boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1}_{|x^{0:t}} \circ \boldsymbol{x}^{t+1}\right).$$

We proceed for each term separately. The expansion of $\boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|x^{0:t}}$ is simply that of $\boldsymbol{r}^{t+1}_{|x^{0:t}}$ multiplied by $\boldsymbol{Q}^{t+1}$, i.e.,

$$\begin{aligned}\boldsymbol{Q}^{t+1}\tilde{\boldsymbol{r}}^{t+1}_{|x^{0:t}} = & \quad \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1} - \alpha \boldsymbol{Q}^{t+1}\textstyle\sum_{0\leq t_1\leq t} \boldsymbol{Q}^{t:t_1}\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right) \\ & \quad + \alpha^2 \boldsymbol{Q}^{t+1}\textstyle\sum_{0\leq t_2<t_1\leq t} \boldsymbol{Q}^{t:t_1}\left\{\left[\boldsymbol{Q}^{t_1-1:t_2}\left(\boldsymbol{r}^{t_2} \circ \boldsymbol{x}^{t_2}\right)\right] \circ \boldsymbol{x}^{t_1}\right\} + \mathcal{O}(\alpha^3) \\ = & \quad \boldsymbol{r}^{t+2} \quad - \alpha \textstyle\sum_{0\leq t_1<t+1} \boldsymbol{Q}^{(t+1):t_1}\left(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}\right) \\ & \quad + \alpha^2 \textstyle\sum_{0\leq t_2<t_1<t+1} \boldsymbol{Q}^{(t+1):t_1}\left\{\left[\boldsymbol{Q}^{t_1-1:t_2}\left(\boldsymbol{r}^{t_2} \circ \boldsymbol{x}^{t_2}\right)\right] \circ \boldsymbol{x}^{t_1}\right\} + \mathcal{O}(\alpha^3).\end{aligned}$$

The second-order expansion (in α) of $\alpha\boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1}_{|x^{0:t}} \circ \boldsymbol{x}^{t+1}\right)$ can be obtained by using the first-order expansion of $\boldsymbol{r}^{t+1}_{|x^{0:t}}$. We have

$$\begin{aligned}\alpha\boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1}_{|x^{0:t}} \circ \boldsymbol{x}^{t+1}\right) &= \alpha\boldsymbol{Q}^{t+1}\left(\left[\boldsymbol{r}^{t+1} - \alpha \sum_{0\le t_1\le t} \boldsymbol{Q}^{t:t_1}\left(\boldsymbol{r}^{t_1}\circ\boldsymbol{x}^{t_1}\right) + \mathcal{O}(\alpha^2)\right]\circ\boldsymbol{x}^{t+1}\right)\\ &= \alpha\boldsymbol{Q}^{t+1}\left(\boldsymbol{r}^{t+1}\circ\boldsymbol{x}^{t+1}\right) - \alpha^2 \sum_{0\le t_1\le t} \boldsymbol{Q}^{t+1}\left\{\left[\boldsymbol{Q}^{t:t_1}\left(\boldsymbol{r}^{t_1}\circ\boldsymbol{x}^{t_1}\right)\right]\circ\boldsymbol{x}^{t+1}\right\} + \mathcal{O}(\alpha^3).\end{aligned}$$

Combining the two expansions yield the desired result. □

We see that the partial expansion follows the same counting intuition as in the static case, with matrices $\boldsymbol{Q}^t$ accounting for plastic particle movements. The first term substracts a fraction α of the plastic particles that have been encountered once (at some time t_1), yet from the locations where they are at time t (which might be different from the location where they were at t_1), as captured by the product $\boldsymbol{Q}^{t:t_1}$. Analogously, the second term adds a fraction α^2 of the plastic particles that have been encountered twice (at times t_1 and t_2). At time t_1, to count the plastic particles already encountered at an earlier time t_2, one needs to account for their potential movements between these two time steps. $\boldsymbol{Q}^{t_1-1:t2}$ map plastic locations at time t_1 to locations at time t_2, hence the Hadamard product between $\boldsymbol{Q}^{t_1-1:t2}(\boldsymbol{r}^{t_2}\circ\boldsymbol{x}^{t_2})$ and $\boldsymbol{x}^{t_1}$.

C Efficient Computations of Plastic Dynamics

In Section 4 we introduce the dynamics of plastic movement and how they are affected by the history collection. In this section, we describe how these predictions of plastic dynamics are stored in practice, and how the inputs to our optimization model can be efficiently computed.

C.1. Data set description

Plastic density data is provided by describing the trajectory of individual plastic 'particles', where a particle is a discretization unit for the fluid mechanical model. This description is referred to as a Lagrangian description. Our one-year plastic data contains the trajectory of $N = 1,212,613$ particles, discretized over T equally-spaced intervals. For example, the original model used a 1-day discretization timestep ($T = 365$), which we extended to a 3-hour discretization timestep via interpolation $T = 365 \times 8$). Eventually, the latitude and longitude of all particles are stored in two $N \times T$ arrays, which we denote as matrices $\boldsymbol{Lo}$ and $\boldsymbol{La}$.

For any location ℓ (defined as an 8×8 km region) and any time t, we can efficiently identify the set of particles present in this location at that time by filtering the rows of $\boldsymbol{Lo}$ and $\boldsymbol{La}$, i.e., we can compute

$$\mathcal{P}_\ell^t := \{n \in [N] \ : \ (La_{n,t}, Lo_{n,t}) \in \ell\},$$

where $(La_{n,t}, Lo_{n,t}) \in \ell$ means that particle n is in location ℓ. Observe that these sets $\{\mathcal{P}_\ell^t\}_{\ell \in \mathcal{L}}$, for a fixed t, can all be computed simultaneously in $\mathcal{O}(N)$ operations, by doing a single pass through all particles $n \in [N]$.

C.2. Computation of raw plastic density map

Based on the data set described above, our first task is to compute the raw plastic density map for the whole year, without considering any history collection. To do so, we can simply count the number of particles in each location at each time, i.e.,

$$r_\ell^t = \left|\mathcal{P}_\ell^t\right| \times \text{weight factor},$$

where the weight factor simply converts a number of particle into a mass of plastic. As discussed above, for a given time t, the entire map $\boldsymbol{r}^t$ can be computed in $\mathcal{O}(N)$ operations. So, we obtain the T maps $\{\boldsymbol{r}^t\}_{t \in [T]}$ in $\mathcal{O}(NT)$ time.

Alternatively, one could have computed $\boldsymbol{r}^0$, in $\mathcal{O}(N)$ time, and the transition matrices $\{\boldsymbol{Q}^t\}_{t \in [T]}$. This is, however, significantly more time consuming.

Indeed, we can compute the element $Q_{\ell,\ell'}^t$ by computing the ratio of particles that move from ℓ to ℓ' between t and $t+1$:

$$Q_{\ell',\ell}^t = \frac{|\mathcal{P}_\ell^t \cap \mathcal{P}_{\ell'}^{t+1}|}{|\mathcal{P}_\ell^t|}.$$

Computing a column of $\boldsymbol{Q}^t$, $\{Q_{\ell',\ell}^t\}_{\ell' \in \mathcal{L}}$ takes $\mathcal{O}(N)$ operations so computing the full matrix takes $\mathcal{O}(N|\mathcal{L}|)$. Accordingly, computing the density maps using the transition probability matrices $\{\boldsymbol{Q}^t\}_{t \in [T]}$ requires $\mathcal{O}(N|\mathcal{L}|T)$ iterations, where $|\mathcal{L}| \approx 10^5$.

C.3. Computation of path-dependent plastic density map

Compared to the raw map, it is more difficult to compute the path-dependent density map. Intuitively, we need to consider the effect of historical collection on the multi-period future. Mathematically, the reward decomposition described in Proposition EC.1 requires multiple computations between $\boldsymbol{Q}^{t:t_1} = \boldsymbol{Q}^t \times \cdots \times \boldsymbol{Q}^{t_1}$, $\boldsymbol{x}$, and α. To solve that, we propose two ways:

1. First, we can pre-compute $\boldsymbol{r}^0$ and $\{\boldsymbol{Q}^t\}_{t\in[T]}$ and then recursively compute $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t}}$ according to (3). This approach requires a number of iteration in the order of $\mathcal{O}(N|\mathcal{L}|T + |\mathcal{L}|^2T)$.

2. Alternatively, we can use the expansion presented in Proposition EC.1 and compute each term sequentially. The zero-order term is simply the original density maps, without any collection, and can be computed in $\mathcal{O}(NT)$ time. Then, for the first-order term, we can compute the term $\boldsymbol{Q}^{t:t_1}(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1})$ directly, without pre-computing the matrices $\boldsymbol{Q}^t$ individually. Indeed, $\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}$ is a vector with only one non-zero element, equal to $|\mathcal{P}^{t_1}_{\ell_1}|$, where ℓ_1 denotes the unique location such that $x^{t_1}_\ell > 0$. Correspondingly, $\boldsymbol{Q}^{t:t_1}(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1})$ represents the vector of locations at time t of plastic particles that were in ℓ_1 at time t_1, i.e.,

$$(\boldsymbol{Q}^{t:t_1}(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}))_\ell = |\mathcal{P}^{t_1}_{\ell_1} \cap \mathcal{P}^t_\ell\}| \times \text{weight factor}.$$

Hence, each correction term $(\boldsymbol{Q}^{t:t_1}(\boldsymbol{r}^{t_1} \circ \boldsymbol{x}^{t_1}))$ can be computed in $O(N)$ operations. Altogether, we can compute the first-order approximation for the path-adjusted rewards in $O(NT)$. We can proceed analogously for the higher-order correction terms, as presented in Section 4.3. However, it becomes prohibitive for $k > 1$. On the positive side, truncating our expansion at an order $k < T$ is guaranteed to provide either a lower or an upper bound on the actual path-dependent reward.

3. Finally, we propose a faster approximation which relies on the efficient particle-level description of plastic dynamics and assigning dynamic weights to the particles. Formally, we assign each particle with a weight $w^t_n(\boldsymbol{x})$ (we will later drop the dependency on $\boldsymbol{x}$ for concision) such that the weight of the particle is divided by α each time it is seen by the system. Mathematically, we compute

$$\begin{aligned} w^0_n(\boldsymbol{x}) &= 1, \quad \forall n \in [N], \\ w^{t+1}_n(\boldsymbol{x}) &= \begin{cases} \alpha w^t_n(\boldsymbol{x}), & \text{if } n \in \mathcal{P}^t_{\ell_t} \text{ with } \ell_t \text{ s.t. } x^t_{\ell_t} > 0, \\ w^t_n(\boldsymbol{x}), & \text{otherwise.} \end{cases} \end{aligned}$$

With these notations, we can count the particles in a given location ℓ by weighting them according to their weight, i.e.,

$$|\mathcal{P}_\ell^t|_{\boldsymbol{w}} := \sum_{n\in[N]\,:\,(La_{n,t},Lo_{n,t})\in\ell} w_n,$$

and approximate the path-dependent plastic density at time t as

$$r^{t+1}_{\ell|\boldsymbol{x}^{0:t}} \approx |\mathcal{P}_\ell^{t+1}|_{\boldsymbol{w}^{t+1}(\boldsymbol{x})} \times \text{weight factor}.$$

The above computations can be efficiently executed in vectorized manner, therefore reducing the computational burden to compute path-dependent reward down to $\mathcal{O}(NT)$.

D Solving the Path-Dependent Longest Path Problem

In this section, we provide further details on how to solve the path-dependent longest path problem (4). We provide a naive mixed-integer optimization (MIO) formulation for Problem (4) in Section D.1. In Section D.2, we describe and compare different strategies for searching for a high-quality solution in Algorithm 2. We provide details on the implementation of our tailored branch-and-bound scheme in D.3. Section D.4 complements the numerical experiments presented in Section 4.6. An alternative strategy for constructing tighter relaxations based on higher-order approximations of the path-dependent rewards is proposed and evaluated in Section D.5.

D.1. MIO formulation

The path-dependent longest path problem (4) can be formulated as a mixed-integer optimization (MIO) problem

$$\max_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{\eta},\boldsymbol{\rho}\geq 0,} \quad \sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}_t}\eta_s^t \quad \text{s.t.} \quad \begin{aligned} &\boldsymbol{\rho}^{t+1} = \boldsymbol{Q}^t\left(\boldsymbol{\rho}^t - \alpha\,\boldsymbol{\eta}^t\right),\\ &\boldsymbol{\eta}^t \leq \boldsymbol{\rho}^t,\\ &\boldsymbol{\eta}^t \leq M\boldsymbol{x}^t,\end{aligned}$$

where the additional variable $\boldsymbol{\rho}^t$ encodes for the vector of path-dependent rewards $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t}}$ and $\boldsymbol{\eta}^t$ for the component-wise product $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t}} \circ \boldsymbol{x}^t = \boldsymbol{\rho}^t \circ \boldsymbol{x}^t$. However, this formulation is notably inefficient to solve for two reasons. First, unlike Algorithms 1-2, it requires the computation of the matrices $\boldsymbol{Q}^t$ for all time periods t, which is time-consuming (see Appendix C for a discussion). Second, it involves a large number of binary variables (the location-level variables x_ℓ^t and the state-level ones $\tilde{x}_s^t$). Even when a network flow formulation is used, integrality constraints cannot be relaxed due to the presence of the additional decision variables $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$.

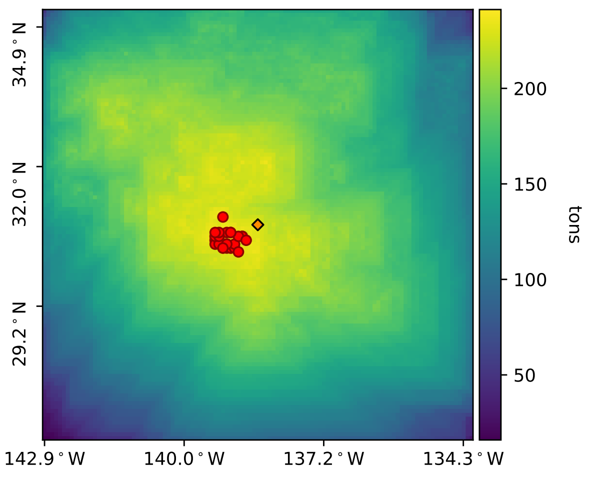
D.2. Comparison of search strategies in Algorithm 2

Recall that in the search-and-bound strategy (Algorithm 2), we search through the terminal states of the DP algorithm (Algorithm 1) to find high-quality trajectories. For a given terminal state s, the terminal value provided by Algorithm 1, $V^T(s)$, can be used as a proxy of the quality of the best trajectory that terminates in state s. However, computing the actual path-dependent reward, i.e., the quantity we are interested in, is time-consuming. Hence, we are interested in identifying a few terminal states $s \in \mathcal{S}_T$ and only evaluating the objective function at these states to make the search more efficient.

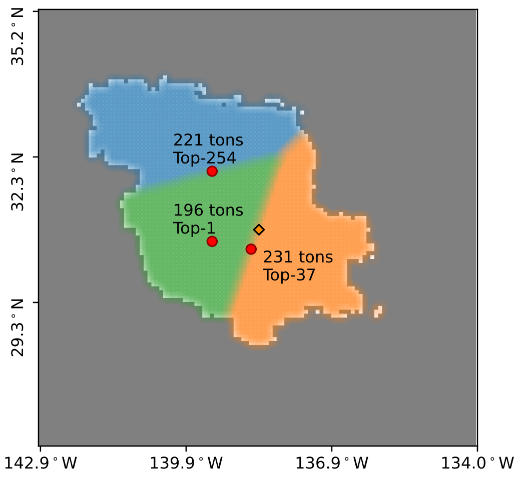
A first strategy is to search through $s \in \mathcal{S}_T$ by decreasing order of $V^T(s)$, as described in pseudo-code below. To control for computational complexity, we can, for example, search through the top-K states only, with K an hyper-parameter controlling the number of path-dependent reward calculations (step ($\star$) in Algorithm 2). Therefore, using a low value of K is critical to ensure numerical scalability. This strategy, however, may compare trajectories that are very similar. For illustration purposes, Figure D.1a displays the map of $V^T(s)$ values on one problem instance (the same instance as that of Figure 5). The red dots indicate the 30 best locations, where, for each location, we consider the highest value of $V^T(s)$, across all terminal states associated with this location. We observe that all top-30 locations are geographically concentrated. Hence, these terminal locations are likely associated with very similar trajectories, leading to very similar path-dependent rewards. In addition, we should keep in mind that one location is associated with multiple states s. So, when searching through the top-K states s, one can compare trajectories that are even more similar (they can end at the same location). All together, while this strategy can improve upon the solution of the DP algorithm (without path-dependency), the improvement will likely be marginal for small values of K.

Alternatively, we use clustering in Algorithm 2 to force a search through states/locations that are geographically dispersed, as displayed in Figure D.1b. For the clustering algorithm, we represent each state by a 2-dimensional vector corresponding to its associated latitude and longitude. In our example, this process inspects the 3 red locations displayed in Figure D.1b. Compared to the top-K approach illustrated in Figure D.1a, this procedure searches through locations that are more dispersed. We represent the path-dependent reward for each inspected location in Figure D.1b. Compared to the trajectory with the highest $V^T(s)$, the best trajectory we found (231 tons) only

```
1  Algorithm 2 – Variant: Top-$K$ search for Algorithm 2
2  Stage 2: Search-and-bound;
3  Initialize upper bound $UB = \max_{s\in\mathcal{S}_T} V^T(s)$;
4  Find $s^\star \in \arg\max_{s\in\mathcal{S}_T} V^T(s)$ and define $\boldsymbol{x}^\star = \boldsymbol{x}^{s^\star}$ ;
5  Initialize lower bound $LB = \langle \boldsymbol{r}_{|\boldsymbol{x}^\star}, \boldsymbol{x}^\star \rangle$, best solution $\hat{\boldsymbol{x}} = \boldsymbol{x}^\star$;
6  Sort $V^T(s)$ by decreasing order ;
7  for $k = 2, \dots, K$ do
8      Get $s^{(k)}$ achieving the $k$-th largest value for $V^T(s)$;
9      Define $\boldsymbol{x} = \boldsymbol{x}^{s^{(k)}}$;
10     $(\star)$ Compute the path-adjusted reward $\boldsymbol{r}_{|\boldsymbol{x}}$;
11     if $\langle \boldsymbol{r}_{|\boldsymbol{x}^s}, \boldsymbol{x}^s \rangle > LB$ then
12         Update lower bound $LB = \langle \boldsymbol{r}_{|\boldsymbol{x}^s}, \boldsymbol{x} \rangle$;
13         Update best solution $\hat{\boldsymbol{x}} = \boldsymbol{x}$;
14     end
15 end
```



(a) Heatmap of the $V^T(s)$ values; The states corresponding to the 30 highest $V^T(s)$ are representated by red dots.



(b) Cluster-K search ($K = 3$) with the 3 selected states represented by red dots.

Figure D.1 Comparing top-K (left panel) and cluster-K (right panel) search strategies on one 7-day planning instance (Jan 15, 2002). The starting location is indicated by a diamond (30.76°N, 138.44°W).

ranked 37th according to its $V^T(s)$ value but increased the weekly collection by 18% (from 231 to 196 tons).

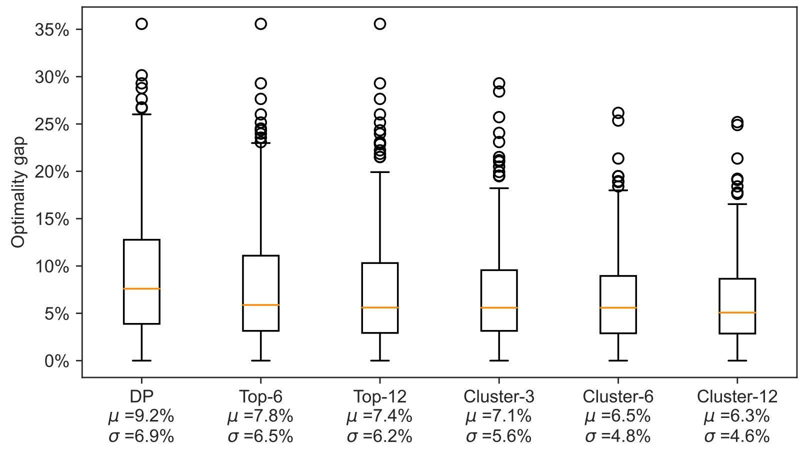


Figure D.2 Distribution of the optimality gap achieved by different search-and-bound approaches: Algorithm 1, top-K search, and cluster-K search. Results are obtained 250 instances of our 7-day routing problem. We also report the mean (μ) and standard deviation (σ) of each distribution.

We now compare both approaches (Top-K and Cluster-K) numerically, on the same $5 \times 50 = 250$ instances of 7-day routing as in Section 4.6 (corresponding to 5 different starting locations, and 50 different starting times). The upper bound is the same for both search strategies and obtained from the DP relaxation. We compute the optimality gap (which is the same as comparing the quality of the solution found) obtained by each approach for different values of K and report their respective box plot in Figure D.2.

As expected, searching through multiple terminal states improves upon the solution of the DP relaxation. We observe that the Cluster-K search strategy is more efficient than the top-K one. Indeed, for the same search budget K, cluster-K achieves lower optimality gaps. For example, Top-12 achieves a 7.4% gap on average, while Cluster-12 achieves 6.3% on average —a 1.1-percentage point (or 17%) improvement. Cluster-3 (7.1% average gap) evaluates 4 times less trajectories than Top-12, yet achieves comparable performance. In addition, we observe that the standard deviation of the gap is smaller for cluster-K (4.6%–5.6%) than for top-K (6.2%–6.5%), indicating that the performances of cluster-K are also more reliable. Therefore, we use the Cluster-K search with $K = 12$ in our implementation.

D.3. Details on our tailored branch-and-bound scheme

D.3.1. Proof of (7) Property (7) is key to our branching scheme. It certifies that we can refine the upper-bound on $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ as we branch. The proof proceeds by double induction.

Proof of (7) We prove the result by induction on k.

For $k = 0$ (no location is fixed), we have $\tilde{\mathcal{X}} = \mathcal{X}$. Furthermore, $\tilde{\boldsymbol{r}}^0 = \boldsymbol{r}^0$, and $\tilde{\boldsymbol{r}}^{t+1}$ and $\boldsymbol{r}^{t+1}$ follow the same recursive formula. So, for any $t \geq 0$, $\tilde{\boldsymbol{r}}^{t+1} = \boldsymbol{r}^{t+1}$. We conclude by noting that $\boldsymbol{r}^{t+1} \geq \boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}$ for any $\boldsymbol{x} \in \mathcal{X}$ (Proposition 1).

Fix $k \geq 0$. We assume that the result holds for k, namely, we assume that for any set $\tilde{\mathcal{X}}$ of the form $\{\boldsymbol{x} \in \mathcal{X} : x^{t_i}_{\ell_i} = 1, \forall i = 1, \dots, k\}$, the reward vector constructed according to the recursive formula (6) satisfies the property (7). Let us consider a set of the form $\tilde{\mathcal{X}} = \{\boldsymbol{x} \in \mathcal{X} : x^{t_i}_{\ell_i} = 1, \forall i = 1, \dots, k+1\}$ and assume, without loss of generality, that $t_1 < \cdots < t_k < t_{k+1}$. Define $\bar{\mathcal{X}} := \{\boldsymbol{x} \in \mathcal{X} : x^{t_i}_{\ell_i} = 1, \forall i = 1, \dots, k\}$, the set obtained by imposing the first k (out of $k+1$) locations imposed by $\tilde{\mathcal{X}}$, and denote $\bar{\boldsymbol{r}}$ the associated reward vector defined by (6).

By the induction hypothesis applied to the set $\bar{\mathcal{X}}$, we have

$$\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \leq \bar{\boldsymbol{r}}^{t+1}, \forall t \geq 0, \quad \forall \boldsymbol{x} \in \bar{\mathcal{X}}.$$

Furthermore, the definition of $\tilde{\boldsymbol{r}}^{t+1}$ only differs from that of $\bar{\boldsymbol{r}}^{t+1}$ after time t_{k+1}. In other words, for any $t \geq 0$, we have

$$\tilde{\boldsymbol{r}}^{t+1} = \begin{cases} \bar{\boldsymbol{r}}^{t+1} & \text{if } t < t_{k+1}, \\ \boldsymbol{Q}^t \left[\bar{\boldsymbol{r}}^t \circ (1 - \alpha \boldsymbol{x}^t)\right] & \text{if } t = t_{k+1}, \\ \boldsymbol{Q}^t \tilde{\boldsymbol{r}}^t & \text{if } t > t_{k+1}. \end{cases}$$

We now analyze each case separately.

For $t < t_{k+1}$: Since $\tilde{\mathcal{X}} \subseteq \bar{\mathcal{X}}$, we have

$$\tilde{\boldsymbol{r}}^{t+1} = \bar{\boldsymbol{r}}^{t+1} \geq \boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}, \quad \forall \boldsymbol{x} \in \tilde{\mathcal{X}} \subseteq \bar{\mathcal{X}}.$$

For $t = t_{k+1}$: Taking $t = t_{k+1} - 1$ in the inequality above, we have just proved that $\tilde{\boldsymbol{r}}^{t_{k+1}} \geq \boldsymbol{r}^{t_{k+1}}_{|\boldsymbol{x}^{0:t_{k+1}-1}}$. Multiplying both sides of these inequalities (component-wise) by $1 - \alpha \boldsymbol{x}^{t_{k+1}} \geq \boldsymbol{0}$ and multiplying by the non-negative matrix $\boldsymbol{Q}^{t_{k+1}}$ yields

$$\tilde{\boldsymbol{r}}^{t_{k+1}+1} := \boldsymbol{Q}^{t_{k+1}} \left[\tilde{\boldsymbol{r}}^{t_{k+1}} \circ (1 - \alpha \boldsymbol{x}^{t_{k+1}})\right] \geq \boldsymbol{Q}^{t_{k+1}} \left[\boldsymbol{r}^{t_{k+1}}_{|\boldsymbol{x}^{0:t_{k+1}-1}} \circ (1 - \alpha \boldsymbol{x}^{t_{k+1}})\right],$$

where the right-hand side is precisely the definition of $\boldsymbol{r}^{t_{k+1}+1}_{|\boldsymbol{x}^{0:t_{k+1}}}$ for trajectories $\boldsymbol{x} \in \tilde{\mathcal{X}}$.

For $t > t_{k+1}$: We have

$$\tilde{\boldsymbol{r}}^{t+1} \geq \boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}}, \forall \boldsymbol{x} \in \tilde{\mathcal{X}}, \quad \text{for } t = t_{k+1}.$$

If the inequality holds for some $t \geq t_{k+1}$, then multiplying by $\boldsymbol{Q}^{t+1} \geq \boldsymbol{0}$ leads to

$$\tilde{\boldsymbol{r}}^{t+2} := \boldsymbol{Q}^{t+1}\tilde{\boldsymbol{r}}^{t+1} \geq \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \geq \boldsymbol{Q}^{t+1}\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \circ (1 - \alpha \boldsymbol{x}^{t+1}) = \boldsymbol{r}^{t+2}_{|\boldsymbol{x}^{0:t+1}},$$

for any $\boldsymbol{x} \in \tilde{\mathcal{X}}$, where the last inequality holds because $(1 - \alpha \boldsymbol{x}^{t+1}) \leq \boldsymbol{1}$. Thus, we can prove the desired properties for all $t = t_{k+1}, \ldots, T$ by induction. □

D.3.2. Implementation details We briefly describe our implementation of branch-and-bound in Algorithm 3.

Algorithm 3: Tailored Branch-and-Bound Algorithm

```
1  Termination criteria: branch_limit, gap_limit, time_limit;
2  Root node:
3  Run Algorithm 1 or 2 to solve (4) over feasible space $\mathcal{X}$ ;
4  Obtain upper-bound $UB(\mathcal{X})$, lower-bound $LB(\mathcal{X})$, and feasible solution $\boldsymbol{x}(\mathcal{X})$ ;
5  Initialize:
6  $UB_{\texttt{global}} \leftarrow UB(\mathcal{X})$; $LB_{\texttt{global}} \leftarrow LB(\mathcal{X})$; $\boldsymbol{x}_{\texttt{best}} \leftarrow \boldsymbol{x}$ ;
7  $active\_nodes \leftarrow [\mathcal{X}]$;
8  for $epoch = 1 : branch\_limit$ do
9      Pop subproblem $\tilde{\mathcal{X}}$ from the list $active\_nodes$ ;              // Node selection
10     From best local incumbent solution $\boldsymbol{x}(\tilde{\mathcal{X}})$, find $(t_0, \ell_0)$ ;        // Branching strategy
11     Define $\tilde{\mathcal{X}}_{\texttt{left}} = \tilde{\mathcal{X}} \cap \{\boldsymbol{x} : x^{t_0}_{\ell_0} = 1\}$, $\tilde{\mathcal{X}}_{\texttt{right}} = \tilde{\mathcal{X}} \setminus \tilde{\mathcal{X}}_{\texttt{left}}$ ;
12     Run Algorithm 1 or 2 to solve (4) over $\tilde{\mathcal{X}}_{\texttt{left}}$ and $\tilde{\mathcal{X}}_{\texttt{right}}$ ;      // Node analysis
13     Push $\tilde{\mathcal{X}}_{\texttt{left}}$ and $\tilde{\mathcal{X}}_{\texttt{right}}$ to the list of $active\_nodes$ ;
14     Update $LB_{\texttt{global}} \leftarrow \max\left\{LB_{\texttt{global}}, LB(\tilde{\mathcal{X}}_{\texttt{left}}), LB(\tilde{\mathcal{X}}_{\texttt{right}})\right\}$ ;
15     Filter $active\_nodes \leftarrow [\bar{\mathcal{X}}$ for $\bar{\mathcal{X}} \in active\_nodes$ if $UB(\bar{\mathcal{X}}) \geq LB_{\texttt{global}}]$;
16     Update $UB_{\texttt{global}} \leftarrow \max\left\{LB(\bar{\mathcal{X}})\right.$ for $\left.\bar{\mathcal{X}} \in active\_nodes\right\}$ ;
17     if $UB_{global} - LB_{global} \leq gap\_limit \times LB_{global}$ OR time elapsed exceeds time_limit then
18         break;
19     end
20 end
```

Node analysis: Given a subset of trajectories $\tilde{\mathcal{X}}$ (which defines a subproblem or a node in the branch-and-bound tree), we apply Algorithm 1 to this restricted problem to obtain both a lower and upper bound (i.e., a feasible incumbent and a bound). We chose not to use Algorithm 2 for time considerations. One could use Algorithm 2 at the root node only, to improve the quality of the first best incumbent solution.

Node selection: Given a list of active subproblems (or nodes), we select the one with the highest upper bound and apply our branching rule. In the presence of ties, we select the node with the highest lower bound (among the ones with the highest upper bounds). Our choice is motivated by the intuition that the loose upper bound is the primary driver of the optimality gap so we prioritize nodes that have the largest upper bound.

Branching strategy: At each step in our branch-and-bound algorithm, we need to select a time t_0 and a location ℓ_0, to further partition the space into the trajectories that go through ℓ_0 at t_0 and those that do not. Given the best incumbent solution (i.e., best trajectory found) at a given node, we select the location-time pair (ℓ_0, t_0) where the raw reward is the highest, i.e., $(\ell_0, t_0) \in \arg\max_{\ell,t} r_\ell^t x_\ell^t$. If there is a tie, we select the location-time pair with the smallest time t_0. On the left child node, our branching improves the approximation of rewards from $x_{\ell_0}^{t_0}$ onwards, so maximizing the raw reward at that point-time could intuitively lead to the biggest correction.

Stopping criteria: We could terminate our branch-and-bound method based on time limit, optimality gap limit, or maximum number of branches created. In our implementation, we only use a limit on the total number of branching operations we do (i.e., $gap\_limit = 0\%$ and $time\_limit = +\infty$ in the pseudo-code).

D.4. Additional numerical evaluation of Algorithms 1, 2, and B&B

Section 4.6 first presents numerical results on 250 instances of our 7-day routing problem. The instances differ in their starting times—50 different options corresponding to the beginning of week 2 to 51—and in their starting locations—5 options, (31.92°N, 142.4°W), (30.72°N, 138.4°W), (34.32°N, 144.0°W), (33.92°N, 140.0°W), (31.92°N, 148.0°W), displayed in Figure D.3.

Table 1 in Section 4.6 reports the average optimality gap obtained by each method. As a complement, Figure D.4 represents the distribution (box plot) of the same optimality gaps. In Figure D.4a, we represent the distribution of the gaps *returned* by Algorithm 1 and Algorithm 2, i.e., using the upper bound from the linear path-independent relaxation (1), while we use the suboptimality gap obtained with the upper bound from B&B Figure in Figure D.4b. We observe that Algorithm 2's performance is better on average, more consistent.

To illustrate how the search procedure in Algorithm 2 impacts the quality of the solution, we focus on one of these 250 instances and represent in Figure D.5 the quantity of plastic collected over the 7 days (56 time periods) by the solution of Algorithm 1 (blue lines) and Algorithm 2 (orange

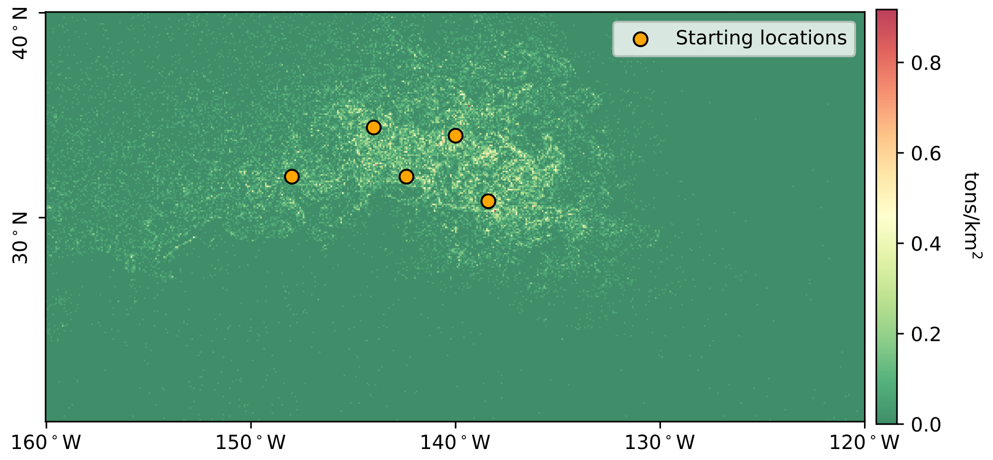
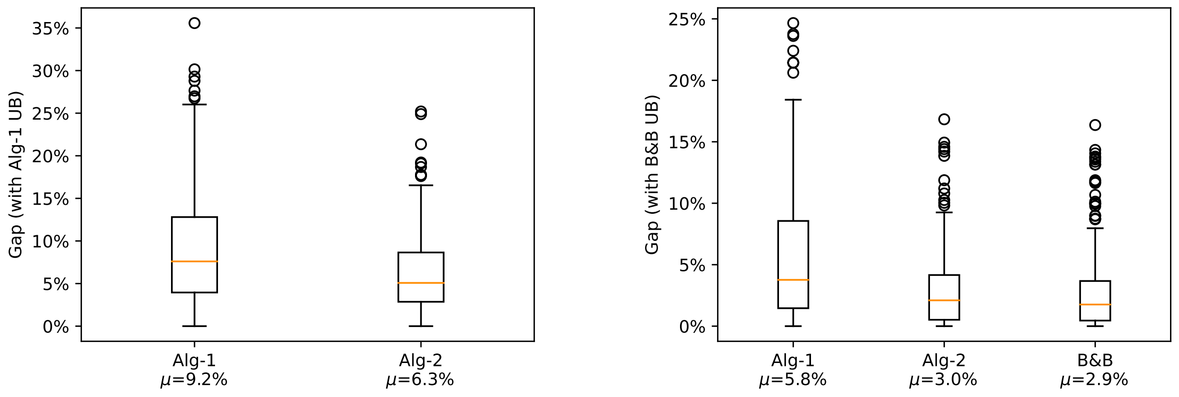


Figure D.3 Five starting locations selected for the experiments in Section 4.6. The color represents the plastic density in each location on day 180 of 2002.



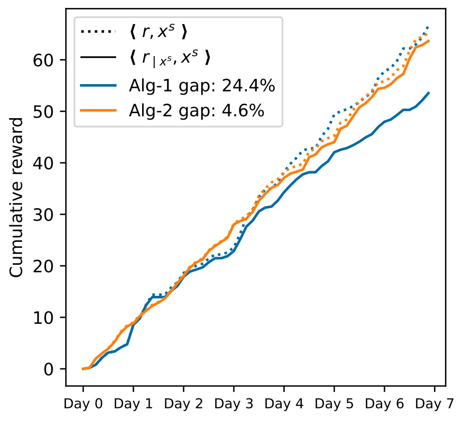
(a) With the UB from (4)

(b) With the UB from B&B

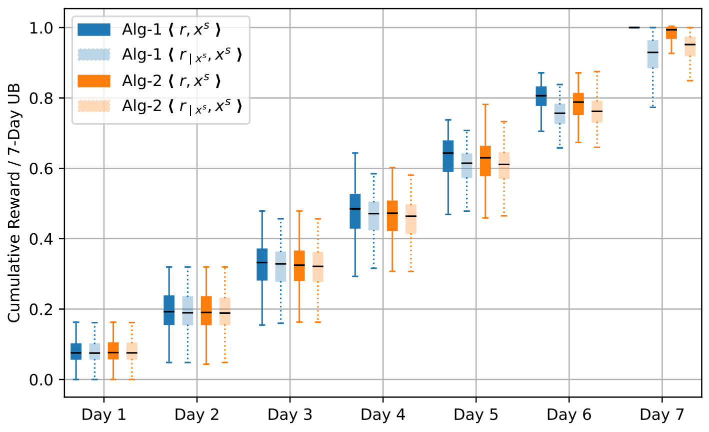
Figure D.4 Distribution (box plot) of the optimality gaps achieved by Algorithm 1, Algorithm 2, and our branch-and-bound algorithm (B&B) across 250 7-day routing instances (with $\alpha = 0.2$).

lines Cluster-12), both when the density vectors are upper bounded by $\langle \boldsymbol{r}, \boldsymbol{x} \rangle$ (path-independent, dashed lines) and adjusted for path-dependency (solid lines). By comparing the dashed lines, we observe that the solution returned by Algorithm 2 is indeed sub-optimal for the path-independent problem. However, when adjusting for path dependency, the reward associated with solution of Algorithm 1 is significantly lower than expected (especially for time periods $t \geq 27$, i.e., Day 3.5+), much more than for the solution of Algorithm 2. Eventually, the solution provided by Algorithm 2 performs significantly better—here, the optimality gap reduces from 24.4% to 4.6%.

Figure D.5b replicates the analysis of Figure D.5a, aggregated over the 250 instances. To ensure a fair aggregation, the cumulative reward for each instance is normalized by the upper bound



(a) One simulation: start at 2002 Jan 15th, (30.72°N, 138.4°W)



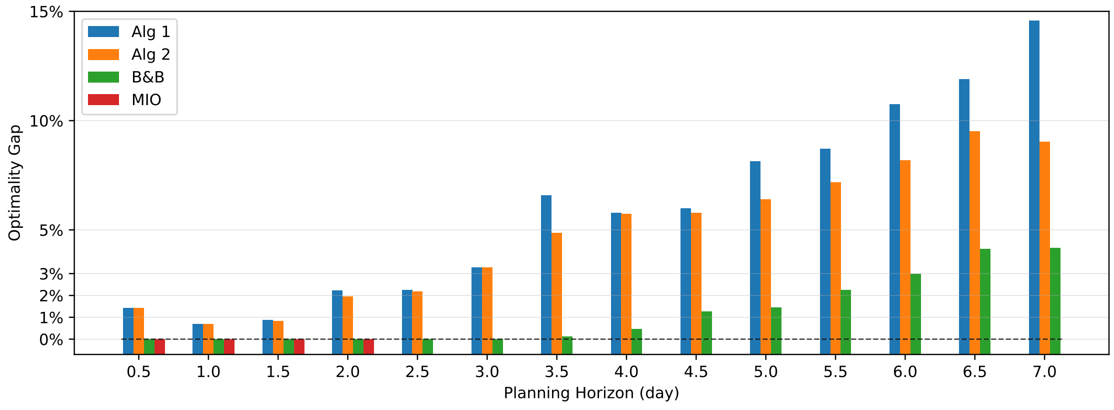
(b) Distribution of all 250 simulations (cumulative rewards at the end of each day)

Figure D.5 Cumulative rewards of Algorithm 1 and Algorithm 2 (with $\alpha = 0.2$).

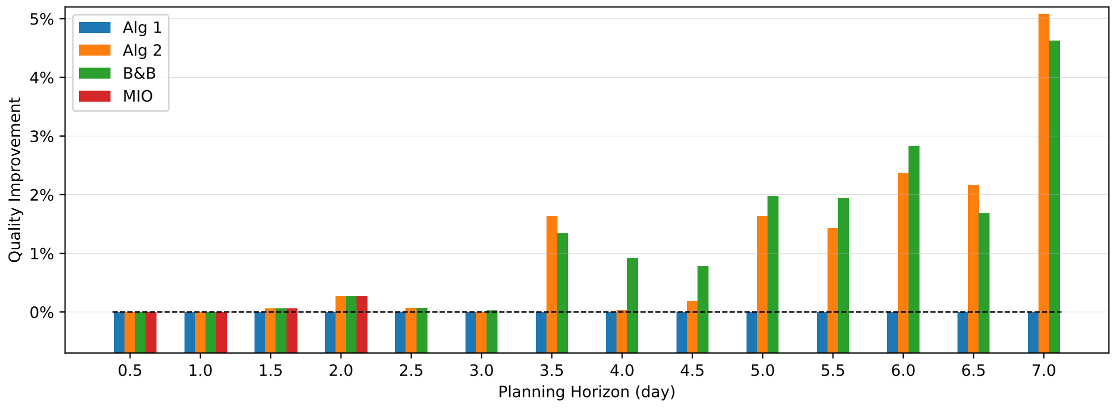
provided by Algorithm 1 on this instance, and we report the distribution (box plot) of these normalized rewards. Like in Figure D.5a, which only includes one simulation, we also observe that the path-dependent performance of Algorithm 1 deviates from its path-independent one. However, unlike Figure D.5a, this deviation is not particularly acute on Day 3, suggesting heterogeneity across instances. Moreover, we observe that the path-dependent reward $\langle \boldsymbol{r}_{|\boldsymbol{x}}, \boldsymbol{x} \rangle$ of Algorithm 2 (dashed light orange boxes) improves upon that of Algorithm 1 (dashed light blue boxes) mostly on the last day. This could be due to the fact that our search strategy in Algorithm 2 clusters the space of trajectories based on their terminal location and selects candidate solution based on their path-dependent reward in the last period.

Section 4.6 also presents results on 6 instances as T increases—from 0.5 days ($T = 4$) to 7 days ($T = 56$). For these instances, we fix the starting point to the center of the GPGP (31.92°N, 142.4°W) and take 6 starting times uniformly spread across the year. For these experiments, we set the maximum number of branching operations for our tailored B&B to 150 (instead of 50 for the 250 7-day instances presented earlier), to better observe the impact of T on the convergence of the algorithm.

Figure 6 compares the computational time required by four methods: Algorithm 1 (Alg 1) Algorithm 2 (Alg 2), our tailored branch-and-bound (B&B), and the MIO formulation (Section D.1). Figure D.6 reports their respective performance in terms of solution quality (top panel) and optimality gap (bottom panel). In terms of optimality gap, we observe in Figure D.6a that the gap returned by Algorithm 1 increases nearly quadratically with T—a simple regression analysis finds



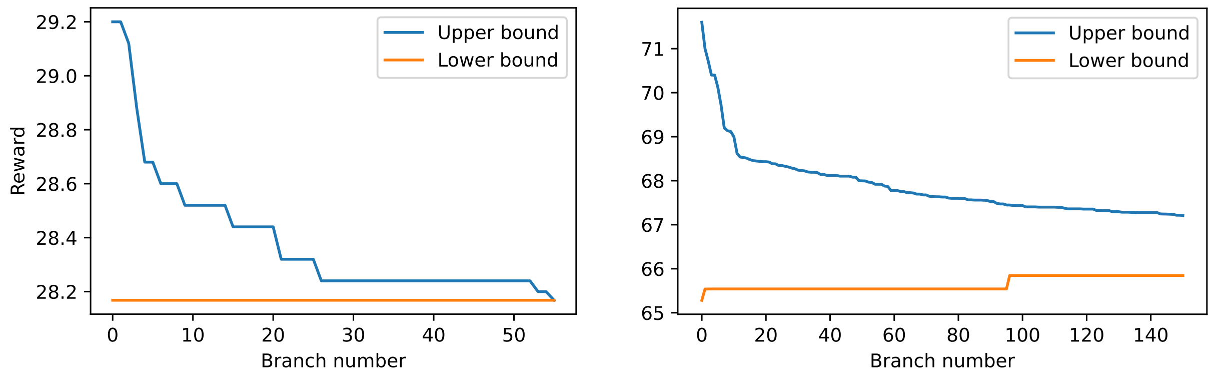
(a) Optimality gap.



(b) Solution quality compared to Alg-1.

Figure D.6 Solution quality and optimality gap achieved by Algorithm 1, Algorithm 2, our tailored B&B, and MIO, averaged over 6 routing instances with increasing planning horizon. For B&B, we impose a limit on the number of branches of 150.

that the gap $\propto T^2$ with an $R^2 = 0.968$—which is consistent with Proposition 1. Algorithm 2 achieves smaller optimality gaps, especially for larger planning horizons (6–7 days) thanks to the improved quality of the solution (Figure D.6b). Furthermore, B&B solves all instances within less than 5% optimality (and within less than 2% of optimality for 5.5-day planning horizons or below). In term of quality (measured in terms of improvement compared with Algorithm 1), we also observe in Figure D.6b that Algorithm 2 returns solutions of similar quality than those returned by B&B solution, for all values of T. Together, these observations suggest that the value of Algorithm 2's optimality gap is mostly indicative of the tightness (or looseness in this case) of the relaxation. The returned solution might be much closer to an optimal solution. Thanks to our tailored B&B, we observe that Algorithm 2 returns an optimal solution on 3-day (or less) instances, and that the true optimality gap is at least half of the returned one for larger instances.



(a) 3 day

(b) 7 day

Figure D.7 Convergence plot of our branch-and-bound algorithm on one simulation (starting on Feb 12th, 2002. from (31.92°N, 142.4°W)).

For illustration purposes, we select one simulation—starting on Feb 12th, 2002, from (31.92°N, 142.4°W)—and present the convergence plot of our B&B algorithm under a 3-day (left panel) and 7-day (right panel) planning horizon in Figure D.7. In both cases, we observe that the primary benefit of B&B is to reduce the upper bound, i.e., to prove optimality. For 3-day planning, an optimal solution is identified at the root node, while branching moderately improves the root node solution for 7-day planning. As expected, the algorithm experiences marginally decreasing returns with most of the UB improvement occurring in the first few iterations. By comparing the two panels, we observe that the number of required nodes increases with T and that the overall converge towards an optimal solution is much slower as T increases. While the algorithm certifies optimality in the day-3 case after branching 55 times, it fails to close the gap after visiting 150 nodes in the day-7 case.

D.5. Alternative relaxations via truncated past trajectories

Recall that our original problem (4) is a polynomial optimization problem where $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ is an order-t polynomial in $(\boldsymbol{x}^0,\dots,\boldsymbol{x}^{t-1})$. In Algorithms 1-2, we relax (4) by upper-bounding the path-dependent reward $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ by a constant (with respect to $\boldsymbol{x}$) vector, $\boldsymbol{r}^t$. We refine this approximation via branching in our tailored branch-and-bound scheme, effectively approximating $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ by a piece-wise constant function.

In this section, we investigate higher-order approximations of $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$. Note that this direction is both a complement and an alternative to our proposal. On one side, we could improve both

Algorithms 1-2 and our branch-and-bound scheme by using the relaxation from an order-k approximation with $k \geq 1$ instead of $k = 0$. On the other side, by considering the sequence of order-k relaxations with $k \to T$, we obtain a scheme that would ultimately converge to an optimal solution without the need for branching.

D.5.1. Example: Order-1 relaxation Let us first illustrate the method using an order-1 approximation. In this case, we assume that the path-dependent rewards only depend on the last location, i.e., the reward at time t only depends on $\boldsymbol{x}^{t-1}$. Formally, we can show the following bound:

Corollary EC.1. *For any time $t \in \mathcal{T}$ and any trajectory $\{\boldsymbol{x}^t\}_{t\in\mathcal{T}}$*

$$\boldsymbol{r}^{t+1}_{|\boldsymbol{x}^{0:t}} \leq \boldsymbol{r}^{t+1} - \alpha\, \boldsymbol{Q}^t \left(\boldsymbol{r}^t \circ \boldsymbol{x}^t\right) \leq \boldsymbol{r}^{t+1}.$$

The first inequality will follow directly from the generic result for the order-k approximation in the next section (Proposition EC.2). The second inequality is a direct consequence of the fact that $\boldsymbol{Q}^t$, $\boldsymbol{r}^t$, and $\boldsymbol{x}^t$ have non-negative entries.

Based on this result, we propose to solve the following optimization problem

$$\max_{\boldsymbol{x}\in\mathcal{X}} \quad \sum_{t\in\mathcal{T}} \left\langle \boldsymbol{r}^t - \alpha\, \boldsymbol{Q}^{t-1}\left(\boldsymbol{r}^{t-1} \circ \boldsymbol{x}^{t-1}\right), \boldsymbol{x}^t \right\rangle. \tag{EC.2}$$

Corollary EC.1 certifies that (EC.2) is a valid relaxation (i.e., upper bound) of (4), which is at least as tight as the path-independent one. However, it is not a linear optimization problem so it cannot be solved by Algorithm 1. Instead, we can extend the definition of the state-space to include both $\boldsymbol{x}^t$ and $\boldsymbol{x}^{t-1}$ in the state variable at time t and solve (EC.2) as a linear longest-path problem in this extended graph.

In the following, we describe how this approach can be generalized to higher-order polynomials and evaluate the numerical performance of the order-1 approximation.

D.5.2. Order-k relaxation Let us denote $\boldsymbol{r}(t,k,\boldsymbol{x})$ the reward at time t obtained by taking the last k locations visited by $\boldsymbol{x}$ only. When $t \geq k$, $\boldsymbol{r}(t,k,\boldsymbol{x})$ should depend on $\boldsymbol{x}^{t-1},\ldots,\boldsymbol{x}^{t-k}$, while, for $t < k$, it can depend on $\boldsymbol{x}^{t-1},\ldots,\boldsymbol{x}^0$ only. We now formally define $\boldsymbol{r}(t,k,\boldsymbol{x})$ and prove that it provides a valid upper bound on $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$.

To define $\boldsymbol{r}(t,k,\boldsymbol{x})$, we start from the path-independent map at time $t-k$ and update it recursively up until time t. Formally, with $k_{\texttt{eff}} := \min(k,t)$, we construct the following sequence

$$\begin{aligned} \boldsymbol{\rho}^{t-k_{\texttt{eff}}+1} &:= \boldsymbol{Q}^{t-k_{\texttt{eff}}}[\boldsymbol{r}^{t-k_{\texttt{eff}}} \circ (1-\alpha \boldsymbol{x}^{t-k_{\texttt{eff}}})], \\ \boldsymbol{\rho}^{t-s+1} &:= \boldsymbol{Q}^{t-s}[\boldsymbol{\rho}^{t-s} \circ (1-\alpha \boldsymbol{x}^{t-s})] \quad \text{for } s = k_{\texttt{eff}}-1,...,1, \end{aligned} \tag{EC.3}$$

and define $\boldsymbol{r}(t,k,\boldsymbol{x}) := \boldsymbol{\rho}^t$. Observe that this recursion defines $\boldsymbol{r}(t,k,\boldsymbol{x})$ as an order-$k_{\texttt{eff}}$ polynomial in $\boldsymbol{x}^{t-1},\dots,\boldsymbol{x}^{t-k_{\texttt{eff}}}$, although the presence of both matrix-vector and Hadamard multiplications does not allow for a compact explicit expression. By construction, we recover the path-dependent reward when $k=t$, i.e., $\boldsymbol{r}(t,t,\boldsymbol{x}) = \boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$. As stated in Corollary EC.1, we have $\boldsymbol{r}(t,1,\boldsymbol{x}) \le \boldsymbol{r}^t$.

We now show that $\boldsymbol{r}(t,k,\boldsymbol{x})$ decreases as the value of k (i.e., the number of past locations taken into account) increases.

Proposition EC.2. *For any $k \le k'$, we have: $\boldsymbol{r}(t,k',\boldsymbol{x}) \le \boldsymbol{r}(t,k,\boldsymbol{x})$ for all $t \in \mathcal{T}$, for all $\boldsymbol{x} \in \mathcal{X}$.*

An important consequence of Proposition EC.2 is that, for any k,

$$\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}} = \boldsymbol{r}(t,t,\boldsymbol{x}) \le \boldsymbol{r}(t,k,\boldsymbol{x}) \le \boldsymbol{r}(t,1,\boldsymbol{x}) \le \boldsymbol{r}^t, \quad \forall t \in \mathcal{T}, \forall \boldsymbol{x} \in \mathcal{X}.$$

In other words, the truncated path-dependent reward $\boldsymbol{r}(t,k,\boldsymbol{x})$ is a valid upper bound of the true path-dependent reward $\boldsymbol{r}^t_{|\boldsymbol{x}^{0:t-1}}$ and is tighter than the raw reward $\boldsymbol{r}^{t+1}$. The more historical decisions included (larger k), the tighter the upper bound is.

Proof of Proposition EC.2 Fix $t \in \{1,\dots,T\}$ and $k \in \{1,\dots,T\}$.

If $k \ge t$, i.e., $k_{\texttt{eff}} = t$ in the recursion (EC.3), then $r(t,k,\boldsymbol{x}) = r(t,t,\boldsymbol{x}) = r(t,k',\boldsymbol{x})$ for any $k' \ge k$ and the result obviously holds.

Let us assume that $k < t$ and let us show that $\boldsymbol{r}(t,k+1,\boldsymbol{x}) \le \boldsymbol{r}(t,k,\boldsymbol{x})$. The result $\boldsymbol{r}(t,k',\boldsymbol{x}) \le \boldsymbol{r}(t,k,\boldsymbol{x}), \forall k' \ge k$ will follow by a simple induction.

We denote $\boldsymbol{\rho}_{(k)}$ and $\boldsymbol{\rho}_{(k+1)}$ the sequence of intermediate maps involved in the recursion (EC.3) defining $\boldsymbol{r}(t,k,\boldsymbol{x})$ and $\boldsymbol{r}(t,k+1,\boldsymbol{x})$ respectively. Because $k<t$, observe that $k_{\texttt{eff}} = \min(k+1,t) = k$ and $k+1$ in these two recursions respectively.

By the initialization of the sequence $\boldsymbol{\rho}_{(k+1)}$, we have

$$\rho^{t-(k+1)+1}_{(k+1)} = \boldsymbol{Q}^{t-(k+1)}[\boldsymbol{r}^{t-(k+1)} \circ (1-\alpha \boldsymbol{x}^{t-(k+1)})] \le \boldsymbol{Q}^{t-(k+1)} \boldsymbol{r}^{t-(k+1)} = \boldsymbol{r}^{t-k},$$

where the inequality holds because $0 \le 1-\alpha \boldsymbol{x}^{t-(k+1)} \le 1$ component-wise and because $\boldsymbol{Q}^{t-(k+1)}$ and $\boldsymbol{r}^{t-(k+1)}$ have non-negative entries. Combining the recursive formula with the above inequality yields

$$\rho^{t-k+1}_{(k+1)} = \boldsymbol{Q}^{t-k}[\boldsymbol{\rho}^{t-k}_{(k+1)} \circ (1-\alpha \boldsymbol{x}^{t-k})] \le \boldsymbol{Q}^{t-k}[\boldsymbol{r}^{t-k} \circ (1-\alpha \boldsymbol{x}^{t-k})] = \rho^{t-k+1}_{(k)}.$$

In other words, we have shown that $\rho_{(k+1)}^{t-s+1} \leq \rho_{(k)}^{t-s+1}$, for $s = k$. Observe that the recursion operator in (EC.3), $\boldsymbol{\rho} \mapsto \boldsymbol{Q}^{t-s}[\boldsymbol{\rho} \circ (1 - \alpha \boldsymbol{x}^{t-s})]$ is non-decreasing in $\boldsymbol{\rho}$, because the entries of $\boldsymbol{Q}^{t-s}$ and $1 - \alpha \boldsymbol{x}^{t-s}$ are non-negative again. So, by an immediate downward induction on s, if follows that

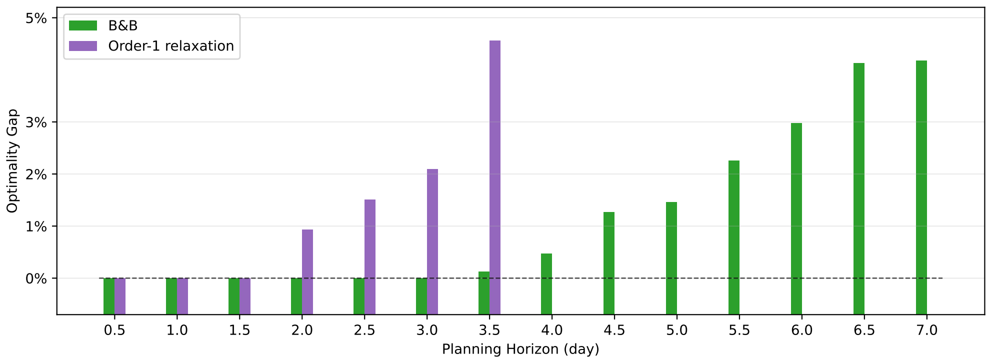
$$\rho_{(k+1)}^{t-s+1} \leq \rho_{(k)}^{t-s+1}, \forall s = k, \ldots, 1.$$

By setting $s = 1$, we have $\boldsymbol{r}(t, k+1, \boldsymbol{x}) = \rho_{(k+1)}^{t} \leq \rho_{(k)}^{t} = \boldsymbol{r}(t, k, \boldsymbol{x})$. $\square$

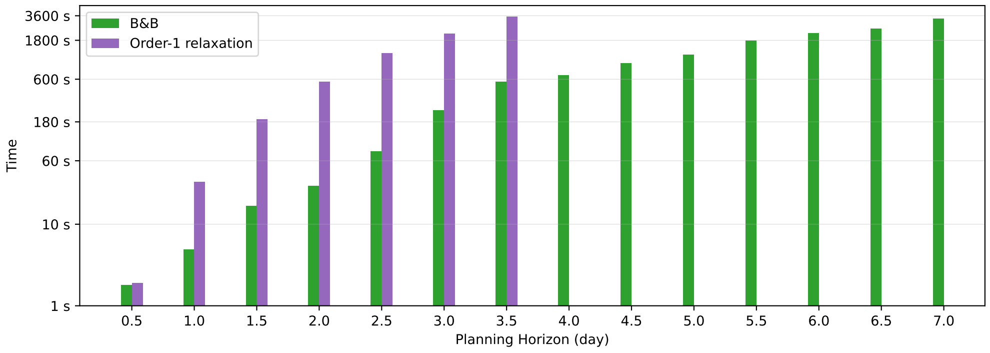
By using Proposition EC.2, we can replace the path-dependent reward $\boldsymbol{r}_{|\boldsymbol{x}^{0:t-1}}^{t}$ in the original problem (4) by the order-k polynomial approximation $\boldsymbol{r}(t, k, \boldsymbol{x})$ and obtain a valid relaxation. To solve it, we need to express this polynomial optimization problem as a longest-path optimization problem over the graph obtained by extending the state space to contain $\boldsymbol{x}^t, \boldsymbol{x}^{t-1}, \ldots, \boldsymbol{x}^{t-k}$. Note that the size of the state space (and of the graph) increases exponentially with k. In the following paragraph, we evaluate the scalability of the order-1 approximation and find it already less scalable that our tailored branching scheme.

D.5.3. Numerical performance of the order-1 relaxation We implement the order-1 relaxation and evaluate its scalability as the planning horizon T increases. To do so, we use the same instances as the ones used in Figures 6 and D.6 to evaluate the scalability of the other algorithms, namely six instances of the routing problem over T time periods, with T increasing from 4 (half a day) to 56 (7 days). We compare the performance (both in terms of optimality gap and computational time) of the order-1 relaxation with that of our tailored branch-and-bound algorithm in Figure D.8.

Overall, we find that the order-1 relaxation does not scale as well as our branch-and-bound when T increases: Although it finds an optimal solution in a reasonable amount of time (less than 3 minutes) for small instances (less than 1.5 days of planning horizon), the order-1 relaxation takes 1 hour to be solved on 3.5-day instances and achieve a 4% optimality gap. On the same instances, our branch-and-bound algorithm terminated in 10 minutes and with a $\approx 0.12\%$ optimality gap on average. With a 1-hour time limit, our branch-and-bound scheme solves 7-day planning instances within 4% optimality.



(a) Optimality gap



(b) Computational time

Figure D.8 Performance of the order-1 relaxation, in terms of optimality gap and computational time, as the planning horizon T increases. For each planning horizon T, metrics are averaged over 6 instances.

E Numerical Experiment Supplement

In this section, we provide supplementary results to the numerical experiments presented in Section 5.

E.1. The Ocean Clean-up's angle heuristic

In Section 5, we compare the performance of our algorithms with that of The Ocean Cleanup's current heuristic. For each direction d, we let $\mathcal{L}_t(\ell, d)$ denote the set of locations that are reachable in t steps from ℓ when looking in the direction of d (here, we associate each direction d with a cone corresponding to all steering angles within $\pm 22.5^\circ$ of d). We associate each direction with a value:

$$\sum_{t \in [T]} \frac{1}{|\mathcal{L}_t(\ell, d)|} \sum_{\ell' \in \mathcal{L}_t(\ell, d)} \frac{r^t_{\ell'}}{\|\ell' - \ell\|_2^2},$$

and pick the best feasible direction according to this criterion. In other words, the benchmark goes in the direction leading to the highest distance-weighted reward. If the system is at capacity but

	Δmean	Std. error	t-stat	p-value	95% Confidence interval	
					Lower	Upper
Myopic vs. Benchmark	21.199	3.032	6.993	1.45×10^{-5}	14.594	27.804
Week vs. Benchmark	21.381	1.940	11.022	1.24×10^{-7}	17.154	25.608
Week-Folding vs. Benchmark	26.422	2.750	9.608	5.51×10^{-7}	20.430	32.413
Week-Rolling vs. Benchmark	26.380	2.154	12.246	3.86×10^{-8}	21.686	31.073

Table E.1 Paired t-test to evaluate the benefit (in terms of weekly plastic collected) of each optimization method compared with the benchmark

the current weather does not allow an extraction, we replace rewards with wave heights in the above formula and steer towards the lowest distance-weighted wave height to start an extraction.

E.2. Overall improvement in plastic collection rate

Figure 7a reports the average weekly quantity of plastic collected by the system, for different routing/extraction scheduling algorithms. We can visually infer that optimization methods provide a significant improvement over the benchmark. To formalize this statement, we conduct a paired t-test to compare the 4-week reward of each method to the benchmark, over the 13 simulations and report the results of these comparisons in Table E.1. Note, unfortunately, that we cannot perform paired t-tests at the week level because the starting location at the beginning of each week is different for each method (while the starting location at the beginning of each simulation is the same).

While the results in the main paper focus mostly on the resulting collection efficiency, we now also report results on computational tractability. Figure E.1 reports the computational time required by each method to plan for one week. For example, for Myopic, it corresponds to the time required for solving seven 1-day routing problems (averaged across the 13 simulations), and for Week, it corresponds to the average time for solving one 7-day problem. As expected, Week-Rolling is the most time consuming method (16.1 minutes), since it solves a 7-day planning problem every day, hence seven times per week. In practice, the route decision can be updated daily, and the time budget for route planning can be up to 12 hours, much longer than our computational time (all methods presented take less than half an hour *per week*). In Figure E.2, we report the same computational times, yet averaged by season instead of over the year. Overall, we observe a milder effect of season on computational time than on performance. Still, bad weather conditions limit the number of feasible states, hence tend to lead to shorter computational times for instances occurring in the winter months than in the summer.

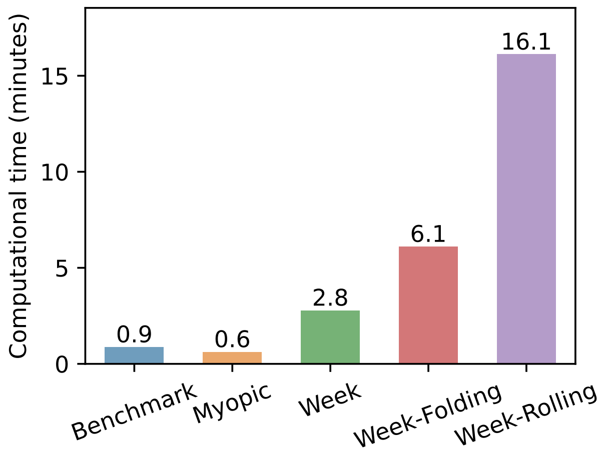
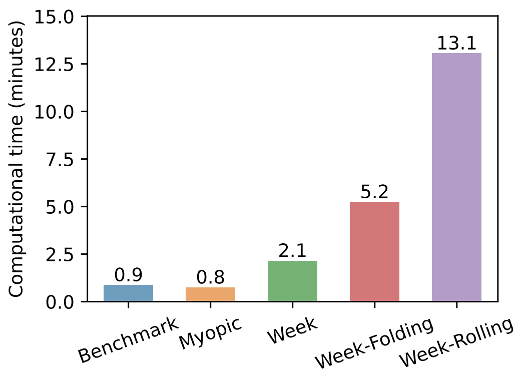
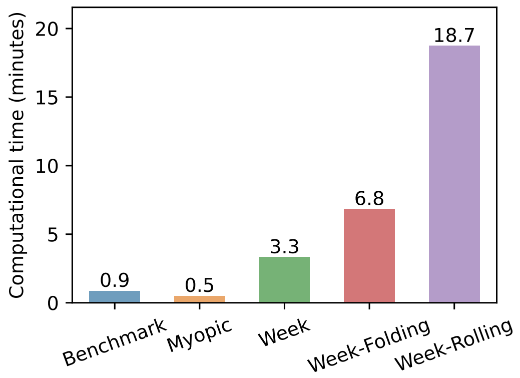


Figure E.1 Average computational time required for 1-week planning, for the benchmark and each of the optimization-based approaches. Results are averaged across the full year (13 simulations).



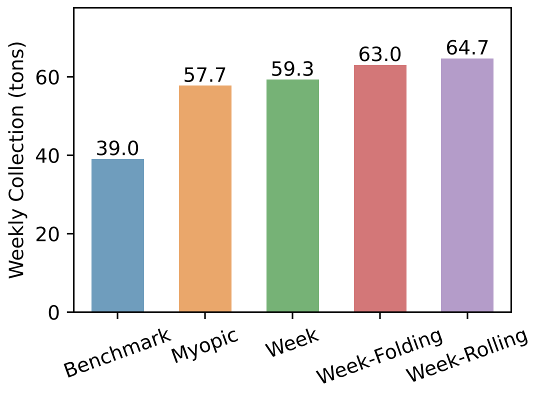
(a) Winter



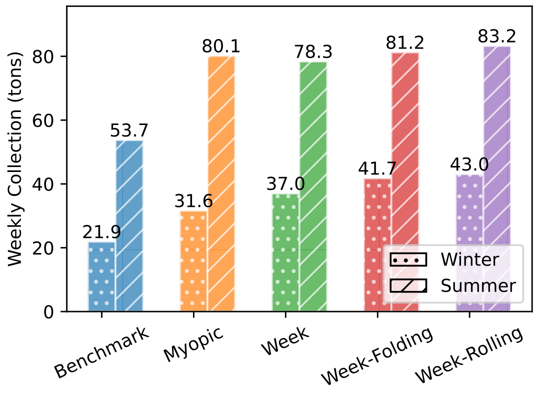
(b) Summer

Figure E.2 Average computational time required for 1-week planning, for the benchmark and each of the optimization-based approaches. Results are averaged across season.

Finally, while Section 4.6 (and Appendix D.4) illustrate the benefit of Algorithm 2 over Algorithm 1 for solving our T-period routing problem and implementing its solution over the T periods, one might wonder what this comparison looks like when implemented with frequent (e.g., daily) resolvings like in Section 5. Accordingly, we ran the same experiments as those for Figure 7, yet with Algorithm 1 instead of Algorithm 2. The results are shown in Figure E.3. Overall, we observe that Algorithm 2 collects more plastic than Algorithm 1, across most implementation variants (myopic, week-folding, week-rolling) by 1.5–4.1%.



(a) Results averaged over the 13 simulations



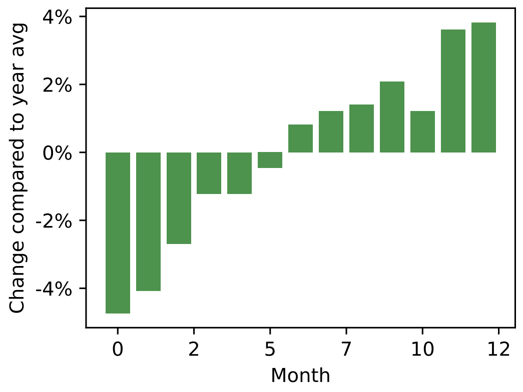
(b) Results grouped by seasons

Figure E.3 Weekly quantity of plastic collected, for the benchmark and each of the optimization-based approaches, using Algorithm 1 instead of Algorithm 2. The experimental setting is the same as for Figure 7.

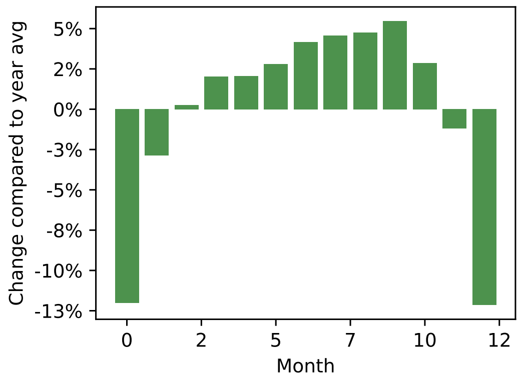
E.3. Heterogeneity across seasons and the impact of extraction scheduling

In Figures 7b and 8a, we observe a significant impact of the season on collection efficiency, for all methods. Note that, in Figure 7b, we group days 1–84 (first three simulations) and day 280–365 (last three simulations) into ‘Winter’ and the remaining 7 simulations as ‘Summer’.

As discussed in Section 5.3, the heterogeneity in efficiency observed cannot be explained by differences in plastic densities. Indeed, as shown in Figure E.4a, real plastic density in the GPGP globally increases over time, due to new plastic being emitted in the oceans. In our data (year 2002), we observe that the average plastic density in the GPGP increased by 10%. However, we do not observe an inverted U-shaped behavior as in Figure 8a, suggesting that weather (and wave height in particular) is the main driver of performance here. To consider the effect of high wave on the collection process, we calculate the average collectable plastic in the GPGP in Figure E.4b, where we treat the plastic density of a high wave location (wave height ≥ 6 meters as zero. It can be seen that the collectable plastic density has such inverted U-shaped, but the difference between the highest collectable density month only exceed 20% more density than the lowest collectable density month. In contrast, in Figure 8a, the maximum gap between the highest collection month and lowest collection month is much higher (more than 300% for Week-Rolling and more than 500% for the Benchmark approach). Therefore, the change in collectable plastic density cannot fully explain the seasonal heterogeneity in performance efficiency.



(a) Real plastic density



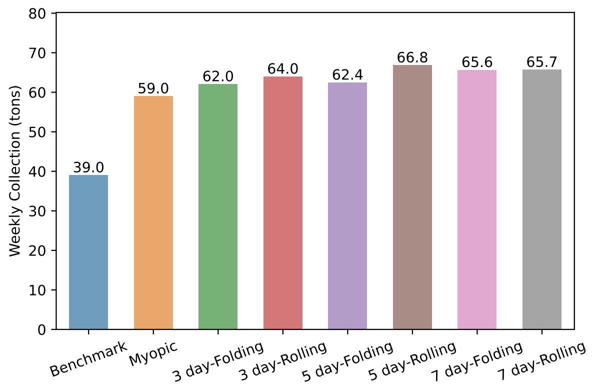
(b) Collectable plastic density

Figure E.4 Average real and collectable plastic density in the GPGP over the year 2002. The collectable plastic density in the right panel is calculated by assigning zero density to any location with wave heights above 6 meters. Values are reported in relative terms compared to yearly average.

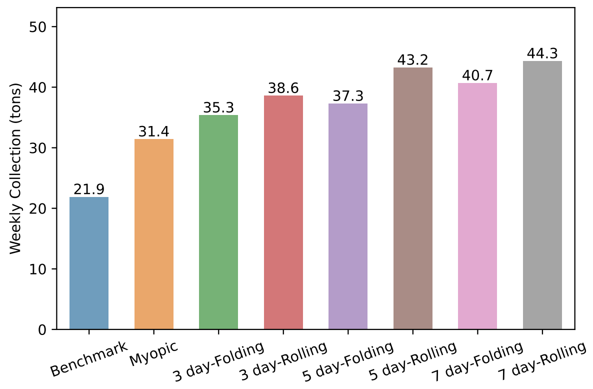
E.4. Impact of planning horizon length on performance

In this section, we investigate how the performance of our algorithms varies with the length of the planning horizon. We run the same experiments as in Section 5.2 for different planning horizons (i.e., 1-, 3-, 5-, and 7-day). All policies are implemented in a rolling and folding horizon fashion, with resolving every day. Results are reported in Figure E.5.

We make the following observations: (i) For all horizon lengths, the rolling horizon implementation dominates the folding one. (ii) Comparing performance across the year and across winter months only, we observe that day-1 (59 tons on average over the year–31.4 tons on average during winter months) is dominated by day-3 (64–38.6), which is in turn dominated by day-5 (66.8–43.2) or day-7 (65.7–44.3). So, performance generally improves as the planning horizon increases. (iii) Using a 5- or 7-day horizon does not materially change the performance, with a small edge for the longer horizon during the winter season. (iv) Smaller horizons seem more sensitive to the season. We believe this is due to the weather conditions required for the extraction. Unlike summer, these constraints are harder to satisfy in winter (see discussion in Section 5.3), so jointly optimizing the routing and the extraction schedule provides a greater edge. On this matter, it is preferable to use a planning horizon long enough to cover the next extraction. Given that the system can stay idle 2–4 days before being able to extract in winter (see Figure 9), we can understand why a planning horizon of 1–3 days might not be sufficient.



(a) Full year 13 instances



(b) Only winter 6 instances

Figure E.5 Weekly quantity of plastic collected, for the benchmark and each of the optimization-based approaches. Standard deviation in parenthesis.